MINING THE RELATION AND IMPLICATION OF USER GENERATED CONTENT IN SOCIAL MEDIA

BY

XIN JIN

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Jiawei Han, Chair
Professor Tarek Abdelzaher
Associate Professor Chengxiang Zhai
Dr. Jie Yu, GE Global Research Center

Abstract

The phenomenal success of social media sites, such as Facebook, Twitter, LinkedIn, Flickr and YouTube, not only revolutionized the way people communicate and think, but also revolutionized the way how corporations do business. During the current age of social media, web usage can be characterized as the decentralization of online information, which now largely consists of high volume and real-time content generated from the bottom-up, where common users are the contributors and producers of information. The transition from Web 1.0 (represented by static webpages instead of dynamic user-generated content) to Web 2.0 (represented by Social Media which consists of large scale of real-time and dynamic user-generated content), makes Internet information become in larger scale, richer, more interactive and complex.

The goal of this thesis is to mine the relation and implication of user generated content in social media. In this thesis, I will present several studies that I have conducted on how to analyze such relation and implication. First, we proposed an approach for similarity computation based on both visual content and link information in social media by a novel way of mutual reinforcement of content similarity and link similarity. Second, we proposed a GAD (General Activity Detection) framework to fully explore the power of activity detection for clustering, which can be used to partition similar content objects into groups. The algorithms (both exact and approximate) developed within this framework can perform fast clustering for large scale content data. Social media content not only relate to each other, but also to outside phenomena and show strong implication with prediction power. In my third work, by aggregating user content information in social

media, we developed a unified model to integrate clustering, ranking and regression for the prediction of stock price change.

To my family and the universe.

Acknowledgments

I would like to thank all the people who have supported and helped me during my Ph.D. study here at UIUC in the past five years.

I would like to express my deepest gratitude to my advisor Professor Jiawei Han for his great support and advise for my Ph.D. study. His vision, enthusiasm, and encouragement inspire me to think more solid and do interesting work. This thesis would not have been possible without his support.

Also I would like to thank other doctoral committee members, Prof. Tarek Abdelzaher, Prof. Chengxiang Zhai and Dr. Jie Yu, for their invaluable help and constructive comments on this thesis.

During my Ph.D. study, it is my great honor to work with talented researchers. I owe sincere gratitude to my collaborators and colleagues, including (but not limited to), Jiebo Luo, Liangliang Cao, Chi Wang, Scott Spangler, Sangkyum Kim, Xide Lin, Hongning Wang, Bolin Ding, Jing Gao, Zhijun Yin, Xiao Yu, Qi He, Quanquan Gu, Manish Gupta, Yizhou Sun, Cuiping Li, Jie Yu, Gang Wang, Andrew Gallagher, Dhiraj Joshi, Ying Chen, Keke Cai, Rui Ma, Mohammad Maifi Hasan Khan, Chandrasekar Ramachandran, Rahul Malik, Yintao Yu, Tianyi Wu, Tim Weninger, Thomas S. Huang, Cedar Pan, Xianyang Zhang, Zhenhui Li, LuAn Tang and Hongbo Deng.

Last but not least, I would like to thank my family for their care and support all the time.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Web 1.0, designed by Tim Berners-Lee and released to the public in 1993, refers to the first stage of the World Wide Web (WWW) which provides a platform of information publishing that is read only. It mainly consists of static and non-interactive web pages made by companies, governments and a few individuals. There were 45 million global users on WWW in 1996.

Web 2.0 is represented by Social Media web applications that facilitate information publishing, sharing, interaction, user-centered design, and collaboration on the World Wide Web. A social media website allows users to create and upload user-generated content and provides convenient ways to let them interact and collaborate with each other, in contrast to Web 1.0 websites where users are limited to the passive viewing of content that was created for them. Examples of social media websites include social networks (Facebook, LinkedIn, MySpace), blogs (Blogger) and microblogs (Twitter, Tumblr), wikis (Wikipedia), forums, reviews and news sharing (Digg), social tagging (or social bookmarking) (Delicious), multimedia sharing (Flickr and YouTube), Social News, prediction markets, virtual worlds, online chatting (AIM, MSN, GTalk, QQ) and social online games. The Facebook website along had over 800 million users worldwide in 2011.

During the current age of social media, web usage can be characterized as the decentralization of online information, which now largely consists of high volume and real-time content generated from the bottom-up, where common users are the contributors and producers of information. The transition from Web 1.0 (represented by static webpages instead of dynamic user-generated content) to Web 2.0 (represented by Social

Media which consists of large scale of real-time and dynamic user-generated content), makes Internet information become in larger scale, richer, more interactive and complex.

The phenomenal success of social media sites, such as Facebook, Twitter, LinkedIn, Flickr and YouTube, not only revolutionized the way people communicate and think, but also revolutionized the way how corporations do business.

The goal of this thesis is to mine the large scale user generated content in social media to explore the hidden relations and implications. Such analysis can be very useful for business, government, as well as individuals.

One basic problem for content relation analysis is content similarity. Content similarity computation is very important for many reasons, for example, grouping together similar content can help better aggregate trends of topics for prediction, identifying similar content objects can help recommendation in social media.

Documents and images are two major content information types in social media. How to compute document similarity and image similarity has been extensively studied in the information retrieval and computer vision areas. However, to compute similarity in such social media especially with image-rich network, is a very useful but also very challenging task, because there exists a lot of information such as text, image feature, user, group and most importantly the *network structure*. Similarity of images is especially hard to estimate, because of the "semantic gap" problem.

We propose an efficient approach called MoK-SimRank to significantly improve the speed of SimRank, which utilizes the network structure to estimate similarity, and introduce its extension HMok-SimRank to work on weighted heterogeneous information networks in social media. Then we propose algorithm IWSL to provide a novel way of integrating both link and content information. IWSL performs content and link reinforcement style learning with either global or local feature weight learning.

Another challenge for social media content mining is the large scale data. To deal with the problem, we propose a GAD (General Activity Detection) framework to fully explore

the power of activity detection for clustering. We design a set of algorithms within this framework for fast clustering in different scenarios. The most important contribution of our work is that GAD is the general solution to exploit activity detection for fast clustering and our algorithms within the framework can achieve very high speed.

Social media content not only relate to each other, but also to outside phenomena and show strong implication for prediction. Both governments and industries are interested in social trends. For example, politicians use polling to measure their popularity for elections and to monitor public sentiment to decide which position to take on social issues. Industry polls potential consumers to understand product acceptance. Although it is an expensive undertaking to perform polling, it is an investment that is critical for organizations both large and small to use for resource allocation and planning.

Social media provides a good platform to explore the global trends and sentiments that can be drawn by analyzing the general patterns of publishing/viewing/sharing content objects in social media. In a sense, for example, each time a content object, such as a comment, image or video, is published or viewed, it constitutes an implicit vote for (or against) the subject of the content. This vote carries along with it a rich set of associated data including time and (often) location information. By aggregating such votes across millions of Internet users, we can develop prediction model to capture the wisdom that is embedded in social media sites for applications such as politics, economics, and marketing.

The rest of the thesis is organized as follows. In Chapter 2, we review related work. Chapter 3 describes our approach for similarity computation of both visual content and link information in a network setting. Chapter 4 presents our approach for conducting fast clustering on large scale data. Chapter 5 describes our unified model to integrate clustering, ranking and regression for prediction based on social media user content data. In Chapter 6, we conclude this thesis.

Chapter 2

Related Work

In this chapter, we review related work in existing literature.

2.1 Similarity Computation for Content Data

Document and images are two major content information types in social media. How to compute document similarity and image similarity has been extensively studied in the information retrieval and computer vision areas. Document similarity methods, such as vector space model and language model, has shown good success in identifying similar documents; however, image similarity is still a very hard task, because of the "semantic gap" problem.

In text-based approach, image similarity is computed by the similarity of the text context, where estimating the similarity of the words in the context is useful for returning more relevant images. WordNet manually groups words into synonym sets, Google Distance [20] computes word similarity by co-occurrence in search results. Flickr Distance [114] considers visual relationship.

In image content-based approach, most methods (such as Google's VisualRank [50]) and systems [106] [100] [37] [38] [87] [35] [15] [69] [109] [75] [54] compute image similarity based on image content features, such as RGB histogram and SIFT.

Hybrid approach combine text features and image content features together [24] [27] [89] [123]. Most commercial image search engines use textual similarity to return semantically relevant images and then use visual similarity to search for visually relevant

images.

Integration-based approaches [27] [89] [123] use linear or non-linear combination of the textual and visual features. However, existing works cannot handle the link structure.

Among algorithms that compute object similarity considering link, SimRank [45] is one of the most popular. It computes node similarity based on the idea that "two nodes are similar if they are linked by similar nodes in the network." In spirit of PageRank [78], SimRank computes the similarity between each pair of nodes in an iterative fashion with a theoretical guarantee of the convergence. There are two problems of SimRank: (1) it is very expensive to calculate; and (2) the similarity is only based on the link information, when consider the images in the network, image similarity can actually also be judged by content features.

2.2 Fast Clustering on Large Set of Content Data

Performing efficient clustering on large content data is especially useful. There are basically two strategies to develop fast clustering on large dataset. One is to develop fast core clustering algorithms, and the other is to develop pre-processing methods, such as sampling, subspace and compression, to reduce the dataset to a smaller size to achieve speedup. For example, CLARA [57] uses sampling strategies to reduce the size of data. BIRCH [128] compresses the original data using CF-tree and then employs the core clustering algorithm (e.g., K-Means) to perform the real clustering. In the thesis, we focus on developing fast core clustering algorithm.

K-Means [66] [70] is one of the most popular clustering algorithms, due to its high efficiency/effectiveness and wide implementation in many commercial/non-commercial softwares. The basic K-Means algorithm performs simple but effective clustering by iteratively partitioning a given dataset into K clusters. For large-scale datasets, the major computation burden of K-Means clustering originates from the numerous distance

calculations between the patterns and the centers [58]. To deal with the problem, fast algorithms with different strategies have been proposed, such as PDS [8], TIE [19], Elkan [9], MPS [86], PAN [79], DHSS [99], FAUPI [62], kd-tree K-Means [81], AKM [83], HKM [76], GT [58] and CGAUCD [63]. Those algorithms come from several research communities, such as data mining, machine learning, pattern recognition, multimedia processing and computer vision.

PDS (Partial Distortion Search) [8] cumulatively computes the distance between the pattern and a candidate center by summing up the differences in each dimension. If the dimensionality is high, PDS may still needs to compute many dimensions to stop accumulation. TIE (Triangular Inequality Elimination) [19] uses the triangle inequality condition for metric distance to prune candidate centers, thus reduces the number of distance calculations. TIE needs extra space $O(K^2)$ to save a distance matrix for the center vectors, and the entries are recalculated at the beginning of each partition. Elkan (by Charles Elkan) [9] is an exact fast algorithm for metric distance by using some metric distance properties. It needs to save the distances between every two centers and recompute them at each iteration. The algorithm only works for metric distance, and is not scalable to large K, the number of clusters, since it requires an additional $O(K^2)$ complexity in both space and time. MPS (Mean-distance-ordered Partial Search) [86] is especially designed for Euclidean distance. MPS is faster than K-Means if the improvement gained from pruning exceeds the overhead caused by sorting. PAN [79] rejects unlikely centers using mean values and variances of an input vector and its two sub-vectors. DHSS (Dynamic Hyperplanes Shrinking Search) [99] uses projection values of input vectors and centers on some dimensions to eliminate unlikely candidate centers. The DHSS algorithm with three projections has the less computing time than PAN. FAUPI [62] is another fast-searching algorithm using projection to reduce the dimension and inequality to reject unlikely codewords.

Many of the above algorithms depend on the metric properties and thus only works

for metric distances. Kaukoranta et al. proposed algorithm GT [58] to utilize point activity for fast clustering and showed that it can further speedup PDS [8], TIE [19] and MPS [86]. Lai et al. proposed algorithm CGAUCD [63] as an extension of GT and demonstrated that combining CGAUCD with MFAUPI (which is an extension of FAUPI [62]) achieves the highest speed. Activity detection, which avoids the metric properties, works for both metric and non-metric distances.

2.3 Prediction Based on Social Media Content Data

James Surowiecki published the book "The Wisdom of Crowds" [98], espousing the idea that under the right conditions, a crowd of non-experts can lead to decisions that are even smarter than the experts within the crowd. The conditions include independence of crowd members, decentralization, diversity and a means for aggregating the judgements of members. For a website with a large user base such as Flickr, all of these conditions are met.

Further, other work shows that the actions of individual Internet users, when properly pooled, can indicate macro trends. There are studies using Search Engine queries for influenza Internet surveillance [21], such as Google Trends [36], search advertisement click through [31], Yahoo search queries [84] and health website access logs [52].

Study on user web access logs from the Healthlink Web site [51] showed that there is a moderately strong correlation between the number of influenza-related article accesses and the CDC surveillance data.

Gunther [31] showed that there is a correlation between the number of clicks on keyword "flu" or "flu symptoms" triggered sponsored link in Google AdSense (appeared for Canadian searchers only) with epidemiological data from the flu season 2004/2005 in Canada.

Specifically in [36], Google search engine queries and data from the Centers for Disease

Control (CDC) are used to find 45 specific search terms that are related to the percentage of influenze related physician visits. This model allows for monitoring influenza rates 1-2 weeks *ahead* of the CDC reports.

The problem of using general search engines is that the original query log is not publicly available and the query trends may become noisy under the impact of news events. For example, as soon as a new product is announced by a major technology company, blogs will begin to report and speculate about the product.

Joshua and Ewan [77] used prediction markets and Twitter to predict a swine flu pandemic. They ”explore the hypothesis that social media such as Twitter encodes the belief of a large number of people about some concrete statement about the world”. Such beliefs are aggregated using a Prediction Market specifically concerning the possibility of a Swine Flu Pandemic in 2009. They show that features extracted from Tweets can reduce the error associated with modeling these beliefs. The approach outperforms baseline methods based purely on time-series information from the Market.

Aron Culotta [22] studied how to detect influenza outbreaks by analyzing Twitter messages. Over 500 million Twitter messages were analyzed from an eight month period. The result showed that by tracking a small number of flu-related keywords, we are able to forecast future influenza rates with high accuracy, with a 95% correlation with national health statistics.

Chapter 3

Mutual Reinforcement based Content Similarity Computation in Social Media

One basic problem for content relation analysis in social media is content similarity. In this chapter, we study the problem of how to conduct efficient and effective similarity computation considering the complex network property of social media.

Social multimedia (photo and video) sharing and hosting websites, such as Flickr, Facebook, YouTube, Picasa, ImageShack and Photobucket, are popular around the world, with over billions of photos uploaded by users. Popular Internet commerce websites such as Amazon are also furnished with tremendous amounts of product-related images. In addition, many images in such social networks are accompanied by information such as owner, consumer, producer, annotations and comments. They can be modeled as heterogeneous image-rich information networks. Figure 3.1 shows an example of the Flickr information network, where images are tagged by the users and image owners contribute images to topic groups. Figure 3.2 shows an Amazon information network of product images, categories and consumer tags.

Computing similarity in such large image-rich information networks is a very useful but also very challenging task, because there exists a lot of information such as text, image feature, user, group and most importantly the *network structure*. In text-based approach, estimating the similarity of the words in the context is useful for returning more relevant images. WordNet manually groups words into synonym sets, Google Distance [20] computes word similarity by co-occurrence in search results. Flickr Distance [114] considers visual relationship. In image content-based approach, most methods (such as Google's VisualRank [50]) and systems [106] [100] [37] [38] [87] compute image similarity

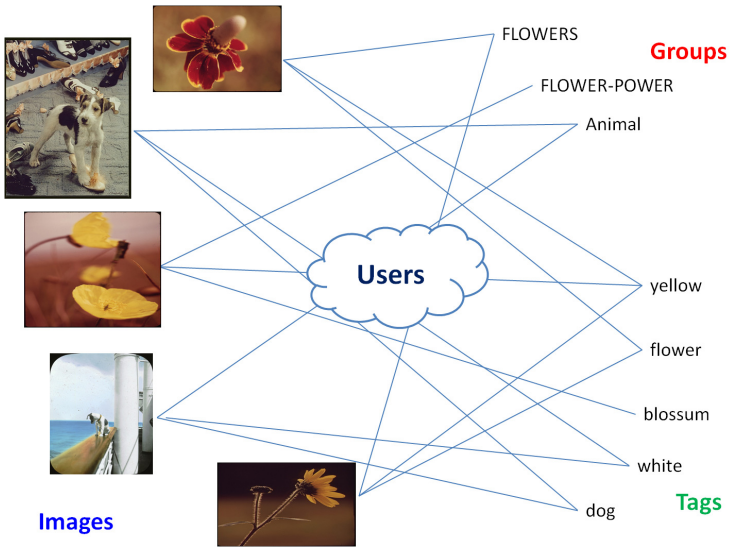


Figure 3.1: Information network for Flickr, connected by images, user tags and groups.

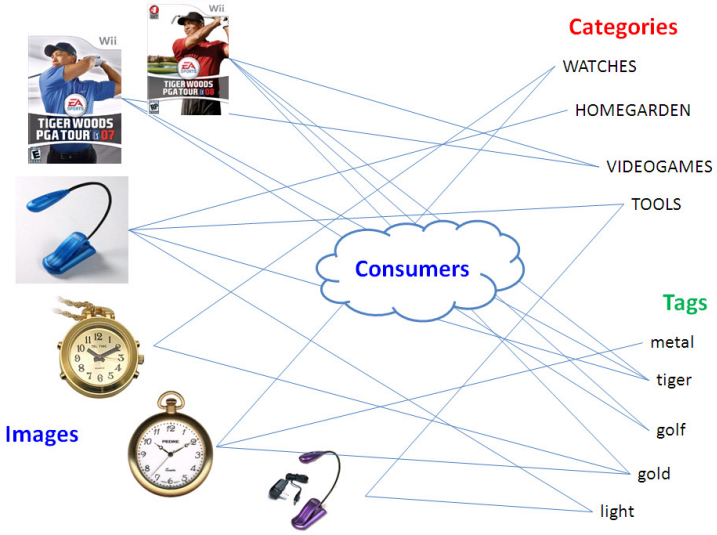


Figure 3.2: Information network for Amazon, connected by products, user tags and categories.

based on image content features. Hybrid approach combine text features and image content features together [24] [27] [89] [123]. Most commercial image search engines use textual similarity to return semantically relevant images and then use visual similarity to search for visually relevant images. Integration-based approaches [27] [89] [123] use linear or non-linear combination of the textual and visual features. However, existing

works cannot handle the link structure. To solve the problem, we propose an image-rich information network model where the similarities between same type of nodes and different types of nodes can be better estimated based on the mutual impact under the network structure.

Among algorithms that compute object similarity in information networks, SimRank [45] is one of the most popular, but it is very expensive to calculate and the similarity is only based on the link information. When consider the images in the network, image similarity can actually also be judged by content features, such as color histogram, edge histogram and SIFT.

We propose an efficient approach called MoK-SimRank to significantly improve the speed of SimRank, and introduce its extension HMok-SimRank to work on weighted heterogeneous information networks. Then we propose algorithm IWSL to provide a novel way of integrating both link and content information. IWSL performs content and link reinforcement style learning with either global or local feature weight learning.

3.1 Preliminaries

We model a *weighted* heterogeneous image-rich information network as a graph $G = (V, E, \mathcal{W})$ with vertices/nodes V, edges E and edge weights $\mathcal{W}$. $V = \{V_q\}$, where $q = 1, ..., Q$ and Q is the number of types of heterogeneous nodes, $|V_q|$ is the number of nodes of type i. Every image has a D-dimensional content feature $F \in R^D$, which is either a single type of feature or a combination of multiple types of features. Add an edge $l_{ij} \in E$ between nodes $i \in V$ and $j \in V$ when they are linked together. Denote ω_{ij} as the weight of this link.

Without losing generality, we consider a heterogeneous network graph with three types of nodes ($Q = 3$): images V_I, groups V_G, and tags V_T. Take Flickr network as an example, there is an undirected link between an image node $e \in V_I$ and a tag node $t \in V_T$ if e is annotated with t, there is also an undirected link between e and a group node $g \in V_G$

if e belongs to group g. There is no link between nodes of the same type.

Table 3.1 lists the major notations.

Table 3.1: Major notations.

Notation	Description
G	graph model of image-rich information network
V	the set of vertices/nodes in the graph
n	$n = \|V\|$, the total number of nodes
V_I	the set of image nodes
V_G	the set of group nodes
V_T	the set of tag nodes
ϖ_{ij}	the weight of a link in the graph.
I	number of iterations for link-based algorithms
$\mathcal{K}(c)$	top k similar candidates of object c
D	number of dimensions for image feature
W	a vector of weights for image feature
C_{ij}^W	weighted content similarity between image i and j
$\mathcal{G}$	global regulated objective function
$\mathcal{L}$	local regulated objective function
$\mathcal{F}_{ij}$	confidence of the link between node i and j
$\mathcal{X}_{ij}$	the χ^2 test statistic distance
$\Omega()$	the sum of the weights

3.2 Fast Link-based Similarity

SimRank [45] is one of the most popular link-based algorithms for evaluating similarity between nodes in information networks. It computes node similarity based on the idea that "two nodes are similar if they are linked by similar nodes in the network." In spirit of PageRank [78], SimRank computes the similarity between each pair of nodes in an iterative fashion with a theoretical guarantee of the convergence. In a basic homogeneous network, SimRank computes the similarity score between two objects o and o' is defined as,

$$S(o, o') = \frac{B}{|N(o)||N(o')|} \sum_{a \in N(o)} \sum_{b \in N(o')} S(a, b) \tag{3.1}$$

with $B \in [0, 1]$ as the damping factor, $N(o)$ as the in-link nodes of o, $|N(o)|$ as the cardinality of set $N(o)$. There are two special cases: (1) if $o = o'$, then $S(o, o') = 1$; and (2) if $N(o) = \varnothing$ or $N(o') = \varnothing$, then $S(o, o') = 0$.

For a network G of n nodes, the memory space needed by SimRank to store the similarity pairs is $O(n^2)$. Denote P as the time spending for calculating Equation 3.1, then the time complexity is $O(n^2P)$ in a single iteration. Because SimRank is computed iteratively, the total time complexity is $O(In^2P)$ for I iterations. The original SimRank is too computationally expensive to be used in large scale networks.

3.2.1 Mok-SimRank for Fast Computation

Some algorithms [45] [65] [33] [108] [124] have been proposed for more efficient SimRank computation. We use the pruning idea proposed by [45]. Initialize every object's top k ($k \ll n$) similar objects as candidates and focus computation on the chosen candidates. This can reduce the space complexity to $O(nk)$ and time complexity to $O(InkP)$, and such method can be denoted as k-**SimRank**. We choose this strategy because it fits the property of large-scale image retrieval wherein most images are dissimilar to the query image and estimating their similarities many times is a waste. The time complexity of P in k-SimRank is $O(|N(o)||N(o')|log(k))$, where $log(k)$ is the complexity to decide whether $N_j(o')$ is a candidate of object $N_i(o)$.

To make k-SimRank even more computationally efficient, we describe an approach called **Mok-SimRank** (minimum order k-SimRank). Denote $\mathcal{K}(c)$ as c's top k similar candidates. Between $N(o)$ and $N(o')$, denote N_{big} as the neighborhood that has larger cardinality and N_{small} as the smaller one.

The basic idea of Mok-SimRank is that beginning with N_{small}, for every $c \in N_{small}$, compute the scores considering two cases:

- Case 1 ($k < |N_{big}|$): For $d \in \mathcal{K}(c)$, check whether $d \in N_{big}$. If true, return the score; else

if false, get zero;

- Case 2 ($k \geq |N_{big}|$): For $d \in N_{big}$, check whether $d \in \mathcal{K}(c)$. If true, return the score; else if false, get zero.

The time complexity of P is then reduced to be always the minimum combination P_{min},

$$P_{min} = \begin{cases} O(|N_{small}|klog(|N_{big}|)) & : \quad if\ k < |N_{big}| \\ O(|N_{small}||N_{big}|log(k)) & : \quad if\ k \geq |N_{big}| \end{cases} \tag{3.2}$$

This is the optimal combination that achieves the minimum cost via automatically making the minimum computation order.

Table 3.2 summarizes the time and space complexity of the link-based similarity algorithms in a homogeneous network of n nodes.

Table 3.2: Complexity of algorithms in *Homogeneous Network*.

Algorithm	**Time Complexity**	**Space Complexity**
SimRank	$O(In^2P)$	$O(n^2)$
K-SimRank	$O(InkP)$	$O(nk)$
Mok-SimRank	$O(InkP_{min})$	$O(nk)$

3.2.2 HMok-SimRank for Weighted Heterogeneous Networks

Mok-SimRank can be extended to work for a *weighted* heterogeneous information network, which contains multiple types of nodes. To explain, we take the image-rich information network from Flickr as an example. Similar images are likely to link to similar groups and tags, so we define the link-based semantic similarity between images $e \in V_I$ and $e' \in V_I$ as follows,

$$S_{m+1}(e, e') = \alpha_I S_m^G(e, e') + \beta_I S_m^T(e, e') \tag{3.3}$$

with

$$S_m^G(e,e') = \frac{B_I^G}{\Omega(N^G(e)) * \Omega(N^G(e'))} \sum_{a \in N^G(e)} \sum_{b \in N^G(e')} \Psi_{ee'}^{ab} S_m(a,b) \tag{3.4}$$

$$S_m^T(e,e') = \frac{B_I^T}{\Omega(N^T(e)) * \Omega(N^T(e'))} \sum_{a \in N^T(e)} \sum_{b \in N^T(e')} \Psi_{ee'}^{ab} S_m(a,b) \tag{3.5}$$

where $N^G(e)$ is set of groups image e links to, $N^T(e)$ is set of tags image e links to. α_I and β_I are the weights of link-based similarity for group and tag, respectively. We set both as 0.5 in experiment to treat them as equally important. B_I^G and B_I^T are the damping factors.

$\Omega(N^G(e))$ is the sum of the weights for the links between image e and nodes in $N^G(e)$,

$$\Omega(N^G(e)) = \sum_{a \in N^G(e)} \varpi_{ea} \tag{3.6}$$

$\Psi_{ee'}^{ab}$ is the importance/contribution of $S(a,b)$ for $S(e,e')$ considering the link weighting, and is defined as the multiplicative combination of the weights of the two links l_{ea} and $l_{e'b}$,

$$\Psi_{ee'}^{ab} = \varpi_{ea} * \varpi_{e'b} \tag{3.7}$$

Weight ϖ can be set manually or automatically. The simplest case is that we set all weights to 1, then the network essentially becomes unweighted and all links are treated as equally important. However, in real applications, the links can be of non-equal importance. Take Amazon as an example, the tag frequency represents the number of users who think the tag is relevant to the product. So we can use the tag frequency (or log value) as weight ϖ_{ea}, for the link between product image e and tag a.

Similarly, we can define and compute the link-based group and tag similarity.

The group similarity is computed via the similarity of the images and tags they link to. For each pair of groups $g \in V_G$ and $g' \in V_G$,

$$S_{m+1}(g,g') = \alpha_G S_m^I(g,g') + \beta_G S_m^T(g,g') \tag{3.8}$$

with

$$S_m^I(g, g') = \frac{B_G^I}{\Omega(N^I(g)) * \Omega(N^I(g'))} \sum_{a \in N^I(g)} \sum_{b \in N^I(g')} \Psi_{gg'}^{ab} S_m(a, b) \tag{3.9}$$

$$S_m^T(g, g') = \frac{B_G^T}{\Omega(N^T(g)) * \Omega(N^T(g'))} \sum_{a \in N^T(g)} \sum_{b \in N^T(g')} \Psi_{gg'}^{ab} S_m(a, b) \tag{3.10}$$

where $N^I(g)$ is the set of images of group g, and $N^T(g)$ is set of tags of group g. The meaning and setting of parameters α_G, β_G, B_G^I and B_G^T are similar to those in Equations 3.3, 3.4 and 3.5.

The tag similarity is calculated via the similarity of the images and groups they link to. For each pair of tags t and t',

$$S_{m+1}(t, t') = \alpha_T S_m^I(t, t') + \beta_T S_m^G(t, t') \tag{3.11}$$

with

$$S_m^I(t, t') = \frac{B_T^I}{\Omega(N^I(t)) * \Omega(N^I(t'))} \sum_{a \in N^I(t)} \sum_{b \in N^I(t')} \Psi_{tt'}^{ab} S_m(a, b) \tag{3.12}$$

$$S_m^G(t, t') = \frac{B_T^G}{\Omega(N^G(t)) * \Omega(N^G(t'))} \sum_{a \in N^G(t)} \sum_{b \in N^G(t')} \Psi_{tt'}^{ab} S_m(a, b) \tag{3.13}$$

where $N^I(t)$ is the set of images tagged by t, and $N^G(t)$ is set of groups tagged by t. The meaning and setting of parameters α_T, β_T, B_T^I and B_T^G are similar to those in Equations 3.3, 3.4 and 3.5.

The similarity score for any pair of nodes within the same type is initialized as 0 for different nodes and 1 for the same node. We do not consider similarity between nodes from different types.

The similarity scores in Equations 3.4, 3.5, 3.9, 3.10, 3.12 and 3.13 can be efficiently computed by the idea introduced in Mok-SimRank. So we can still achieve high efficiency via Mok-SimRank for heterogeneous networks. *Note* that the computation of the similarity scores in Equations 3.3, 3.8 and 3.11 are mutually dependent.

Generally, basic SimRank, K-SimRank and Mok-SimRank can be extended to compute link-based similarity in (weighted) heterogeneous networks, and we call them **HSimRank** and **HK-SimRank** and **HMok-SimRank** to distinguish with the situation in homogeneous networks.

Table 3.3 summarizes the time and space complexity of the link-based similarity algorithms in a (weighted) heterogeneous network of p types of nodes, with $m_i (i \in \{1, ..., p\})$ nodes for type i. Note that the total number of nodes in the network is $n = \sum_{i=1}^{p} m_i$.

Table 3.3: Complexity of algorithms in (weighted) *Heterogeneous Network*.

Algorithm	**Time Complexity**	**Space Complexity**
HSimRank	$O(I \sum_{i=1}^{p} m_i^2 P)$	$O(\sum_{i=1}^{p} m_i^2)$
HK-SimRank	$O(I \sum_{i=1}^{p} m_i k P)$	$O(\sum_{i=1}^{p} m_i k) = O(nk)$
HMok-SimRank	$O(I \sum_{i=1}^{p} m_i k P_{min})$	$O(\sum_{i=1}^{p} m_i k) = O(nk)$

3.3 Weighted Content-based Similarity

Image similarity can be estimated from image content features [24] [26] [122], such as color histogram, edge histogram [68], Color Correlogram [42], CEDD [18], GIST, texture features [2] [73] , Gabor features [80] [92], shape [43] [60]and SIFT [67].

We represent an image as a point in a D-dimension feature space with either a single type of feature or a combination of multiple types of features. Tang at el. proposed strategies to integrate both local and global features [101]. If the integrated feature space has fixed number of dimensions, our approach is also applicable.

Normalize feature $F \in R^D$, where D is the number of dimensions in the feature space, to be of unit length: for any f^d, the value of feature F on dimension d ($d = 1, ..., D$), divide it by the sum of values on all dimensions.

$$f^d = f^d_{orig} \Big/ \sum_{d=1}^{D} f^d_{orig} \tag{3.14}$$

The χ^2 test statistic distance between two feature vectors F_i and F_j is defined as:

$$\mathcal{X}_{ij} \equiv \mathcal{X}(F_i, F_j) \equiv \sum_{d=1}^{D} c_{ij}^d \tag{3.15}$$

$$= \frac{1}{2} \sum_{d=1}^{D} \frac{(f_i^d - f_j^d)^2}{f_i^d + f_j^d} \tag{3.16}$$

When feature vectors are normalized to unit length, the χ^2 test statistic distance varies from 0 to 1, with 0 indicating the most similar and 1 indicating the most different.

Existing studies on metric learning [120] [116] [113] have empirically and theoretically shown that instead of treating each dimension of the feature vector equally, a learned weighted metric can significantly improve the performance for tasks such as image retrieval [120], classification [6] and clustering [119]. This is because the feature dimensions are usually not equally important for measuring the similarity between images. We can obtain better result by putting more weights on a subset of features, which are more relevant to the semantic meaning of the images.

Based on the χ^2 test statistic distance and a D-dimensional feature weighting vector $W = (w^1, w^2, ..., w^D)$, we define the weighted content similarity C_{ij}^W between images i and j as follows:

$$C_{ij}^W \equiv 1 - \frac{1}{2} \sum_{d=1}^{D} \frac{(w^d f_i^d - w^d f_j^d)^2}{w^d f_i^d + w^d f_j^d} \tag{3.17}$$

$$= 1 - \sum_{d=1}^{D} w^d c_{ij}^d \tag{3.18}$$

which is used to evaluate the image similarity. There are two reasons why we choose χ^2: Firstly, it has shown performance as good as, and sometimes better, with cosine, $L1$ and $L2$ measures for image similarity in our experiments by human judgment. It has been used in image retrieval [10] [107] and obtains best accuracy as a kernel for SVM-based image classification [104] [46] [127]. Secondly, its sum of square formula makes it convenient to

perform Gradient Decent-based optimization used in our algorithm as described in the next section.

3.4 Reinforced Integration of Link and Content Similarities

Using content similarity only may lead to unsatisfying results. Figure 3.3 shows one pair of Flickr images that have similar content similarity estimated from the low-level feature, but with different semantic meaning.

Figure 3.3: Images with high visual similarity, but low semantic similarity.

Direct use of link information solely based on human annotations may also lead to unsatisfying results if the annotation is wrong, too general, or incomplete. In addition, if the image does not link to any object in the information network, then only based on link information cannot work.

Figure 3.4 shows several examples that are all linked to tag ”flower” but they are not visually similar.

Traditional thinking is to combine content and link information to achieve more robust performance. In this section, we describe a novel way to integrate these two types of information.

Figure 3.4: Images annotated by the tag "flower", but with low visual similarity.

3.4.1 Learning the Feature Weights

To build a bridge between the content and semantics, we learn a weighting vector $W \in R^D$ for the feature space to force the weighted content-based similarity C_{ij}^W to be somehow consistent with the semantic link-based similarity S_{ij}. We consider two types of approaches: global feature learning and local feature learning.

Global Feature Learning (GFL) We minimize the following regulated objective function to find W,

$$\mathcal{G}(W, p, h) = \beta||W||^2 + \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} \mathcal{Y}_{ij} \tag{3.19}$$

where $|V_I|$ is the number of images and $\mathcal{K}(i)$ is the set of top k neighboring candidate images of image i, by combining both top $k/2$ most visually similar and top $k/2$ most semantically similar images. The first component $\beta||W||^2$ is a L_2 regulation. The second component $\mathcal{Y}_{ij}$ serves as the bridge between content-based similarity and link-based similarity,

$$\begin{aligned} \mathcal{Y}_{ij} &= (C_{ij}^W - (pS_{ij} + h))^2 \mathcal{F}_{ij} \\ &= (1 - \sum_{d=1}^{D} w^d c_{ij}^d - (pS_{ij} + h))^2 \mathcal{F}_{ij} \end{aligned} \tag{3.20}$$

where C_{ij}^W (defined in Equation 3.17) and S_{ij} (defined in Equation 3.3) may have different scales and shifts, so we introduce parameters p and h to automatically estimate them.

If the tags (or groups) of an image are incomplete (0 or very few) and thus cannot fully describe its semantic meaning, the link-based similarity becomes less reliable. In order to consider this factor, we introduce $\mathcal{F}_{ij}$ as the confidence (or importance) of S_{ij}, and define it as a function of the number of linked annotations (including both tags and groups) to the images i and j,

$$\mathcal{F}_{ij} = 1 - e^{-\tau \Gamma_{ij}} \tag{3.21}$$

where Γ_{ij} is defined as the minimum number of links (including tags and groups) for image i and j, i.e.,

$$\Gamma_{ij} = min\Big\{|N^T(i) + N^G(i)|, |N^T(j) + N^G(j)|\Big\}. \tag{3.22}$$

Parameter τ rescales the value of Γ_{ij} to adjust the importance of count. It can be either manually set or automatically estimated by the average of Γ_{ij} values.

The value of $\mathcal{F}_{ij}$ varies from 0 to 1, with 1 indicating the highest confidence and 0 indicating the lowest confidence. If $\Gamma_{ij} = 0$ (i.e., at least one of images does not have any tag and group), then $\mathcal{F}_{ij} = 0$, which means the link-based similarity is not reliable because the semantic information given by human annotation is missing.

Note that we can also use graph ranking algorithms, such as PageRank [78] and HITS [17], to estimate the importance of a node, instead of simple counting. In addition, we may also consider the $tf * idf$ score of tags, similar to document retrieval, to give higher weights to topic terms and lower weights to general terms.

The overall importance $\mathcal{F}_S$ is defined as

$$\mathcal{F}_S = \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} \mathcal{F}_{ij} \tag{3.23}$$

By considering $\mathcal{F}_S$, we obtain the following new global objective function,

$$\mathcal{G}(W, p, h) = \beta ||W||^2 + \frac{1}{\mathcal{F}_S} \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} \mathcal{Y}_{ij} \tag{3.24}$$

where β is the weight of the regulator.

To find (W^*, p^*, h^*) that minimizes the above objective function, we first compute the first-order partial derivatives,

$$\frac{\partial \mathcal{G}}{\partial w^d} = 2\beta w^d + \frac{1}{\mathcal{F}_S} \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} 2\mathcal{F}_{ij}(1 - \sum_{d=1}^{D} w^d c_{ij}^d - (pS_{ij} + h))(-c_{ij}^d) \tag{3.25}$$

$$\frac{\partial \mathcal{G}}{\partial p} = \frac{1}{\mathcal{F}_S} \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} 2\mathcal{F}_{ij}(1 - \sum_{d=1}^{D} w^d c_{ij}^d - (pS_{ij} + h))(-S_{ij}) \tag{3.26}$$

$$\frac{\partial \mathcal{G}}{\partial h} = \frac{1}{\mathcal{F}_S} \sum_{i=1}^{|V_I|} \sum_{j \in \mathcal{K}(i)} 2\mathcal{F}_{ij}(1 - \sum_{d=1}^{D} w^d c_{ij}^d - (pS_{ij} + h))(-1) \tag{3.27}$$

The variables are estimated by Gradient Decent (or Stochastic Gradient Decent, which could be faster) iteratively,

$$w_{m+1}^d = w_m^d - \gamma_{w^d} \frac{\partial \mathcal{G}}{\partial w^d}\bigg|_{w^d = w_m^d} \quad for\ all\ d \in \{1, ..., D\} \tag{3.28}$$

$$p_{m+1} = p_m - \gamma_p \frac{\partial \mathcal{G}}{\partial p}\bigg|_{p = p_m} \tag{3.29}$$

$$h_{m+1} = h_m - \gamma_h \frac{\partial \mathcal{G}}{\partial h}\bigg|_{h = h_m} \tag{3.30}$$

where m denoted the m-th iteration.

After global feature learning, based on the new feature weighting, update the image similarity as a combination of content-based and link-based similarity,

$$S(i, j) = (1 - \mu)C_{ij}^{W^*} + \mu(p^* S_{ij} + h^*) \tag{3.31}$$

where parameter $\mu \in [0, 1]$ controls the weight of link-based similarity in the combination. We could manually set a value based on user preference or automatically use the value of $\mathcal{F}_{ij}$ which estimates the confidence score of the link-based similarity, as defined in

Equation 3.21.

Local Feature Learning (LFL) The problem of global feature learning is that using a global feature weighting for all images may be too general. Different images may belong to different semantic topics and thus need different weightings to capture their specific important features. Therefor, we can perform local feature learning (LFL) to find a specific feature weight W_i^* for image i. More specifically, for each image i, we learn a local weight W_i, which minimizes the following objective function $\mathcal{L}_i$,

$$\begin{aligned} \mathcal{L}_i(W,p,h) &= \beta||W||^2 + \frac{1}{\mathcal{F}_S^i}\sum_{j\in\mathcal{K}(i)} \mathcal{Y}_{ij} \\ &= \beta||W||^2 + \sum_{j\in\mathcal{K}(i)} \frac{(C_{ij}^W - (p_i S_{ij} + h_i))^2 \mathcal{F}_{ij}}{\mathcal{F}_S^i} \end{aligned} \tag{3.32}$$

where $\mathcal{F}_S^i$ is the normalization factor to sum up the importance of all pairs for image i, and is defined as

$$\mathcal{F}_S^i = \sum_{j\in\mathcal{K}(i)} \mathcal{F}_{ij} \tag{3.33}$$

Similar to GFL, in order to find parameters (W^*, p^*, h^*) for image i that minimizes its objective function, we first compute the first-order partial derivatives,

$$\frac{\partial \mathcal{L}_i}{\partial w^d} = 2\beta w^d + \frac{1}{\mathcal{F}_S^i}\sum_{j\in\mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^W - (pS_{ij} + h))(-c_{ij}^d) \tag{3.34}$$

$$\frac{\partial \mathcal{L}_i}{\partial p} = \frac{1}{\mathcal{F}_S^i}\sum_{j\in\mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^W - (pS_{ij} + h))(-S_{ij}) \tag{3.35}$$

$$\frac{\partial \mathcal{L}_i}{\partial h} = \frac{1}{\mathcal{F}_S^i}\sum_{j\in\mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^W - (pS_{ij} + h))(-1) \tag{3.36}$$

Use Gradient Decent to compute parameters iteratively using the following formula,

$$w_{m+1}^d = (1 - 2\gamma_{w^d}\beta)w_m^d + \gamma_{w^d}\frac{1}{\mathcal{F}_S^i}\sum_{j\in\mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^{W_m} - (p_m S_{ij} + h_m))c_{ij}^d \tag{3.37}$$

$$p_{m+1} = p_m + \gamma_p \frac{1}{\mathcal{F}_S^i} \sum_{j \in \mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^{W_m} - (p_m S_{ij} + h_m))(S_{ij}) \tag{3.38}$$

$$h_{m+1} = h_m + \gamma_h \frac{1}{\mathcal{F}_S^i} \sum_{j \in \mathcal{K}(i)} 2\mathcal{F}_{ij}(C_{ij}^{W_m} - (p_m S_{ij} + h_m))) \tag{3.39}$$

Because the weights have similar properties, we set all γ_{w^d} as the same: γ_w. Then the number of γ parameters is reduced from $D + 2$ to 3. We initialize the variables as follows, $w_0^d = 1$ (for all $d = 1, ..., D$), $p_0 = 1$ and $h_0 = 0$.

After local metric learning, based on the learned local weights, the similarity between two images i and j can be computed in two ways: symmetric and asymmetric.

The asymmetric similarity $S^A(i, j)$ is defined as

$$S^A(i, j) = (1 - \mu)C_{ij}^{W_i^*} + \mu(p_i^* S_{ij} + h_i^*) \tag{3.40}$$

where (W_i, p_i, h_i) are the learned parameters for image i.

The symmetric similarity $S^S(i, j)$ is defined as

$$S^S(i, j) = \frac{S^A(i, j) + S^A(j, i)}{2} \tag{3.41}$$

3.4.2 Integration Algorithm

We present novel algorithm to integrate link-based and content-based similarities. A basic approach would be in two-stage: firstly perform HMok-SimRank to compute the link-based similarities and secondly perform feature learning (either GFL or LFL) considering the link-based similarity to update the feature weights, and then update the node similarities based on the new content similarity. Algorithm 1 describes the procedure of the Two-Stage approach.

In the Two-Stage approach, image content similarity is not used to help upgrade tag and group similarities. To solve this problem, we need an algorithm that can provide

Algorithm 1 Two-Stage Approach

Require: G, the image-rich information network.

1. Find top K similar candidates of each object;
2. Initialization;
3. Iterate {
4. Compute link similarity for all image pairs;
5. Compute link similarity for all group pairs;
6. Compute link similarity for all tag pairs;
7. } until converge or stop criteria satisfied;
8. Perform feature learning to update $W = W^*_{m+1}$;
9. Update image similarities by Equation 3.31 (global), or 3.40, 3.41 (local);

Ensure: S, pair-wise node similarity scores.

deeper integration between content and link information. The idea is that after feature learning, we update the image similarity by combining the link similarity with weighted content similarity. (Before going to the next step, one option is to use the new similarity to update the set of similar candidates by introducing more candidates that may have been missed in the initialization step.) Based on the new image similarity, we can update the group and tag similarity. With new tag and group similarity, we can update new link-based image similarity and learn a new weight. The image/tag/group similarities will be mutually updated iteratively until the process converges or any stop criteria is satisfied. We call this approach Integrated Weighted Similarity Learning (IWSL). The term *weighted* has two meanings: (1) we learn weighted content feature, and (2) the algorithm works in a general weighted heterogeneous network. Algorithm 2 describes the procedure of IWSL.

Table 3.4 summarizes the time and space complexity of Two-Stage and IWSL in a (weighted) heterogeneous network of p types of nodes, where there are $m_i (i \in \{1, ..., p\})$ nodes for type i and the total number of nodes is $n = \sum_{i=1}^{p} m_i$. Let J be the number of iterations for Gradient Decent iterative updating, then $Q = O(J|V_I|k)$ is time complexity for feature learning (both GFL and LFL are the same). Denote v as the number of variables to be estimated. For GFL, $v = D + 2$, and for LFL, $v = |V_I|(D + 2)$.

Algorithm 2 Integrated Weighted Similarity Learning (IWSL)

Require: G, the image-rich information network.

1. Construct *kd*-tree [12] (or LSH [23] and *cv*-tree [13] index) over the image features;
2. Find top k (or ϵ-range) similar candidates of each object;
3. Initialize similarity scores;
4. Iterate {
5. Calculate the link similarity for image pairs via HMok-SimRank;
6. Perform feature learning to update $W = W^*_{m+1}$, using either global or local feature learning;
7. (Optional) Search for new top k similar image candidates based on the new similarity weighting;
8. Update the new image similarities $S_{m+1}(i, i')$ by Equation 3.31 (global), or 3.40, 3.41 (local);
9. Compute link-based similarity for all group and tag pairs via HMok-SimRank;
10. } until converge or stop criteria satisfied.

Ensure: S, pair-wise node similarity scores.

Table 3.4: Complexity of Two-Stage and IWSL in *Heterogeneous Network*.

Algorithm	**Time Complexity**	**Space Complexity**
Two-Stage	$O(J\|V_I\|k + I(\sum_{i=1}^{p} m_i k P_{min}))$	$O(nk + v)$
IWSL	$O(IJ\|V_I\|k + I\sum_{i=1}^{p} m_i k P_{min}))$	$O(nk + v)$

3.5 Experiments

Experiments were conducted on a PC with Intel Pentium(R) D 3.4GHz CPU and 4GB RAM, running Windows XP.

3.5.1 Datasets

We conduct experiments on two datasets: **Flickr** and **Amazon**. The Flickr dataset is created by downloading the images and related meta-data information, such as groups and tags using Flickr API. The Amazon dataset is created by downloading product images and related meta-data information, such as category, tags and title, via the API of Amazon. The Amazon API only returns the top 5 tags for each product, so we use the words in the title as additional tags. Product category is treated as group. Table 4.3 shows the statistics for the two datasets.

Table 3.5: Statistics for the datasets

Datasets	Images	Groups	Tags	Links
Flickr	14,559	24	23,420	298,376
Amazon	118,426	41	56,914	1,307,092

For image feature extraction, we extracted CEDD [18], which is a compact descriptor that considers both color and edge features. In literature, it has shown good performance compared with many traditional features. Note that our model is general to other features or combination of them.

Tag Preprocessing. We change all tags to lower case. Tags with only number characters are removed. Stop-words, such as *the*, *you* and *me*, are also removed. The remaining tags are stemmed using the Porter stemming algorithm [85]. We remove very infrequent tags and only retain those that appear in more than k (e.g., k=2) images.

Figure 3.5 shows the tag frequency (the number of images annotated with the tag) of dataset Flickr. We can observe that many tags have low frequency, and a very few tags have high frequency. The Amazon dataset has similar curve shape, as shown in Figure 3.6.

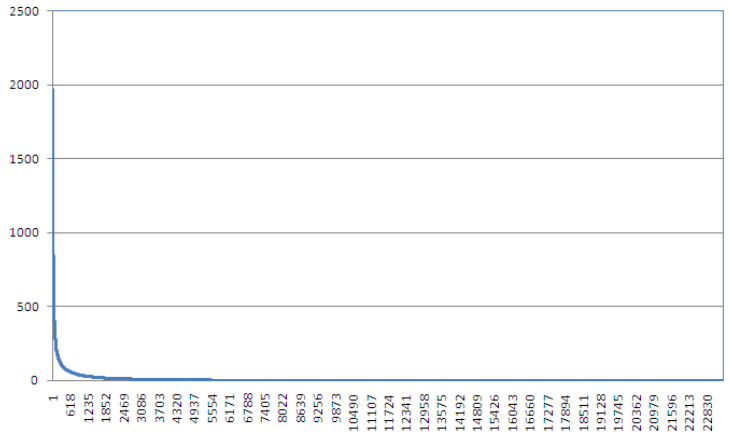


Figure 3.5: Tag frequency for the Flickr dataset. Y-axis denotes the frequency, X-axis denotes the ordered (by frequency) tag id.

Table 3.6 shows the top 10 most frequent tags for the two datasets.

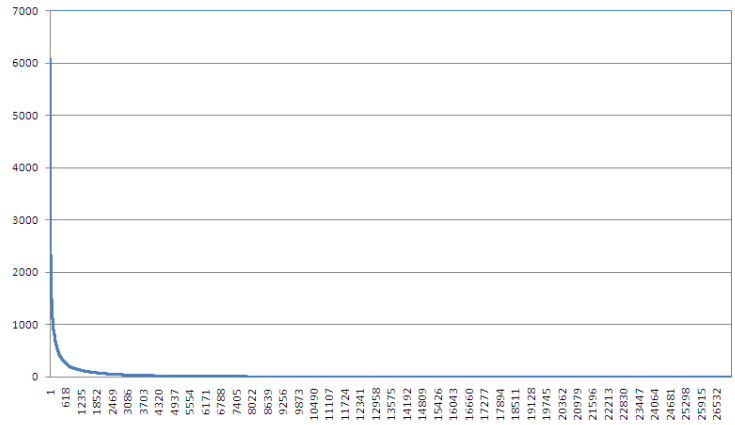


Figure 3.6: Tag frequency for the Amazon dataset. Y-axis denotes the frequency, X-axis denotes the ordered (by frequency) tag id.

Table 3.6: Top 10 most frequent tags.

(a) Flickr

Tag	Frequency
flower	1970
nikon	1545
night	1491
white	1468
black	1288
canon	1252
geotag	1250
light	1231
sky	1127
nature	1122

(b) Amazon

Tag	Frequency
black	6088
pack	4621
case	4235
watch	3987
women	3706
men	3611
classic	3348
music	3321
set	3231
game	3228

3.5.2 Performance Results

Figure 3.7 shows the speed-up ($T_{baseline}/T_i$) of *HK*-SimRank and HMoK-SimRank over baseline HSimRank. With the increase of the number of images, they become increasingly faster than HSimRank. Note that we only show result for as most 3000 images due to the expensive time complexity of the baseline HSimRank, which takes too long for larger data. Figure 3.8 shows the speed-up of HMoK-SimRank over *HK*-SimRank. We can see that HMoK-SimRank is much faster than HK-SimRank. Because IWSL is based on HMoK-SimRank, it has similar time efficiency except for the time spent on feature weight

learning.

For exact executing time, we take the Amazon dataset as an example: when the number of images is 3000, HSimRank takes 598 seconds, *HK*-SimRank takes 24 seconds and HMoK-SimRank takes 9 seconds; when the number of image increase to 20000, *HK*-SimRank takes 928 seconds, while HMoK-SimRank only takes 132 seconds.

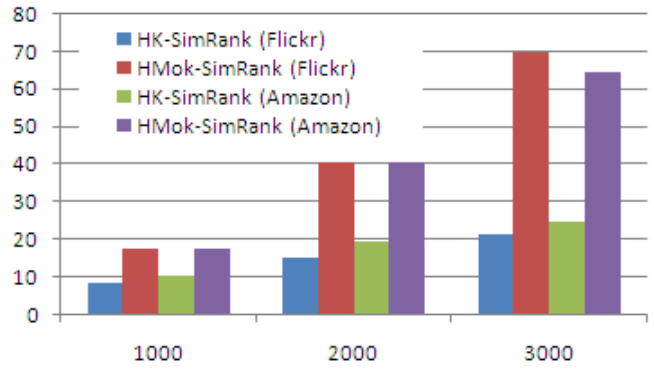


Figure 3.7: Speed-up of HK-SimRank and HMok-SimRank over HSimRank. X-axis denotes the number of images, Y-axis denotes the speed-up (in times ratio).

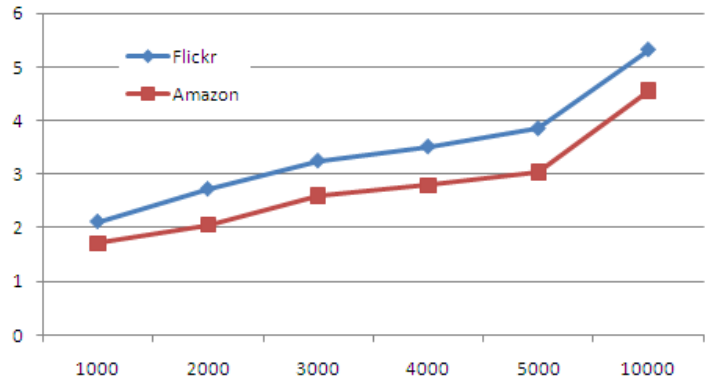


Figure 3.8: Speed-up of HMok-SimRank over HK-SimRank. X-axis denotes the number of images, Y-axis denotes the speed-up (in times ratio).

Because of the large number of images, it is difficult to check one by one to obtain a complete set of relevant images for each query image. In order to generate an approximate ground truth for performance evaluation, we assume that if two images are relevant, their visual similarity should be above a threshold ε_v and the number of shared tags should

also be above a threshold ε_t.

We ignore images which contain less than 5 tags. Such images make up 6.5% and 15.3% of the Flickr and Amazon dataset, respectively. Since all considered images have more than 5 tags, which are human annotations, if two images don't have any common tag, it is likely that the users who made those tags do not think they are relevant. On the Internet, there are many images do not have tags, to simulate the real world case, we randomly select 50% images to remove all their tags. Our algorithm can still find some of them as relevant because we are able to learn a feature weight based on those images which have tags or other link-based information.

We compare VLWC (weighted combination of visual and link similarity without feature weight learning), IWSL_L (IWSL with local feature weight learning) and IWSL_G (IWSL with global feature weight learning) to several baselines: Visual (only use the visual similarity), Text (only use the textual similarity, following a popular text retrieval approach: cosine measure based on the $tf * idf$ weighted tag vector), VTWC [27] (weighted combination of visual and textual similarity, we choose equal weight), Link (HMok-SimRank which only use the link similarity), MinFusion and MaxFusion [27].

Evaluation method: we use mean average precision (MAP) to measure the retrieval performance of the algorithms. For every image in the dataset, we obtain a ranking list of relevant images computed by each algorithm and compute the average precision based on the approximate ground truth before removing tags. The final MAP score for each algorithm is calculated as the mean average precision of each image. Note that there is no training data. So all the algorithms are un-supervised.

Figures 3.9 and 3.10 show the result on Flickr and Amazon data, respectively. We can see that link-based similarity performs better than text-based similarity; VLWC achieves better performance than traditional algorithms by linearly combining visual and link information together. Algorithm IWSL further improves the performance by introducing a new way of integrating content and link information via mutual reinforcement with

feature learning. IWSL_L achieves better results than IWSL_G, because IWSL_L performs local feature learning, which can find a specific and better feature weighting for each image than global feature learning, which finds a general feature weighting for all images.

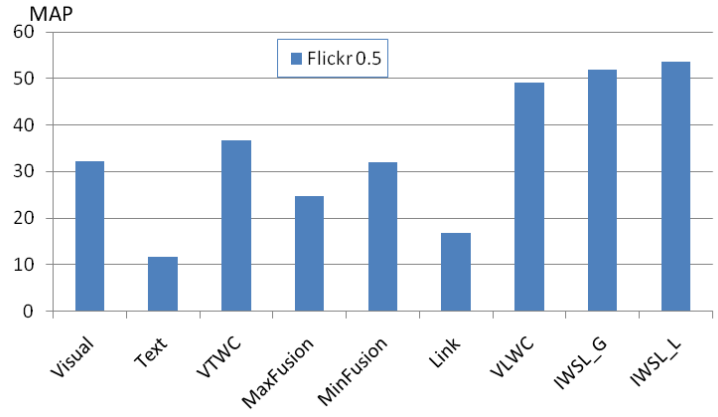


Figure 3.9: MAP of the algorithms on Flickr data. X-axis denotes the algorithms. Y-axis denotes the MAP (%).

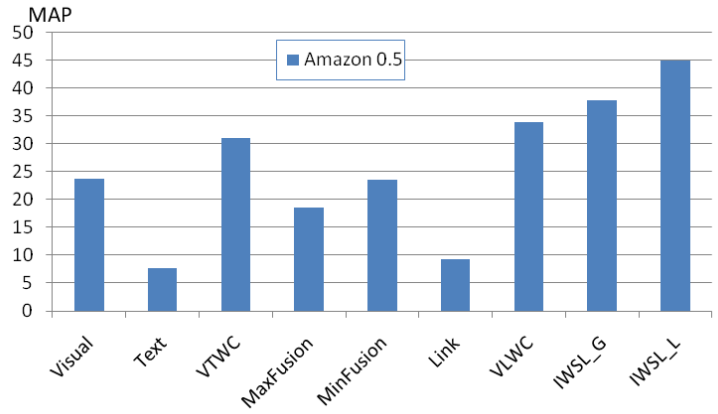


Figure 3.10: MAP of the algorithms on Amazon data. X-axis denotes the algorithms. Y-axis denotes the MAP (%).

Case Study:

As an example from the Flickr dataset, Figure 3.11 shows the top 10 most similar images for a query image about ”moon,” using link-based (SimRank) (1st row), content-based similarity (2nd row), and IWSL (3rd row), respectively. The top left image is the

query image. Clearly, IWSL obtains the most relevant matches for both semantic and visual appearances.

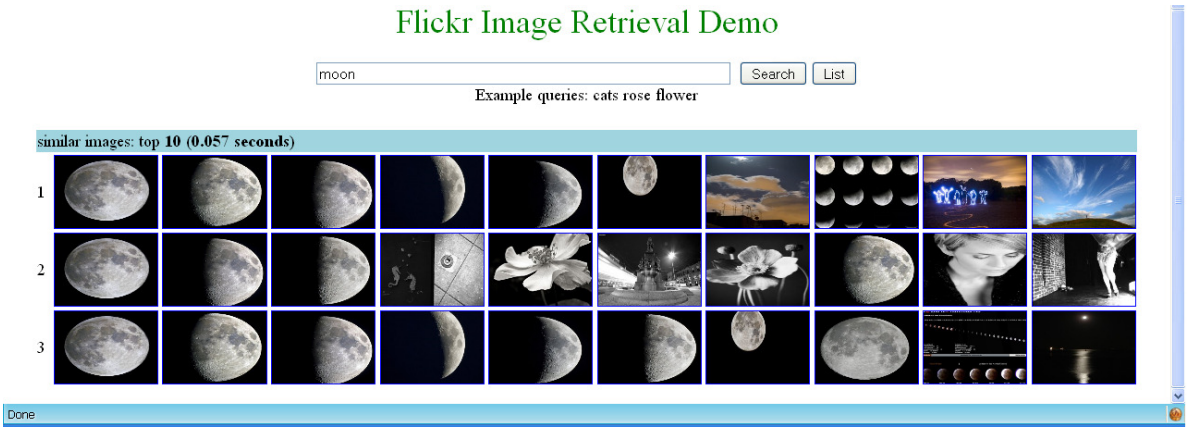


Figure 3.11: Top 10 retrieval results by (1) Link (SimRank), (2) Content similarity, and (3) IWSL. The top left image is the query image from the Flickr dataset. It is tagged with "moon, lune, sky" and belongs to group "After Dark - Night Photography."

In another example from the Amazon dataset, Figure 3.12 shows the top 10 most similar images for a query image about "iPhone," using link-based similarity (SimRank) (1st row), content-based similarity (2nd row), and IWSL (3rd row). Again, IWSL obtains the best results in terms of the relevance for both semantic and visual appearances.

Figure 3.12: Top 10 retrieval results by (1) Link (SimRank), (2) Content similarity, and (3) IWSL. The top left image is the query image from Amazon. It is tagged with "iPhone, invisible shield, accessories" and belongs to category "WirelessAccessories."

Our experiments also show good performance of our algorithm to find similar groups

and relevant tags. For example, similar groups about *night* are "Night Images," "No-Flash Night Shots," "After Dark - Night Photography" and "Night Lights." Relevant tags to flower are "floral," "flora" and "botani." We can use such tag similarity to help find more relevant images for a keyword query.

Parameter Setting The experiments are based on the following parameter setting: $\tau = 0.5$ (Eq. 3.21), $\mu = \mathcal{F}_{ij}$ (Eq. 3.31 and 3.40), $\beta = 0.5$ (Eq. 3.24 and 3.33) and all γ parameters as 0.5 for Gradient Decent. The damping factor parameters defined in Section 3.2.2 are set as 0.8 by following the standard of SimRank.

As an example to demonstrate how we obtain the optimal parameter setting, Figure 3.13 shows the MAP of algorithm IWSL_L w.r.t. parameter β, for both datasets. When β is too small which means significantly ignoring the regulator, the performance is not good, which proves the benefit of introducing the regulator; when β is too big, the performance is also not good, because the regulator will dominate the optimization.

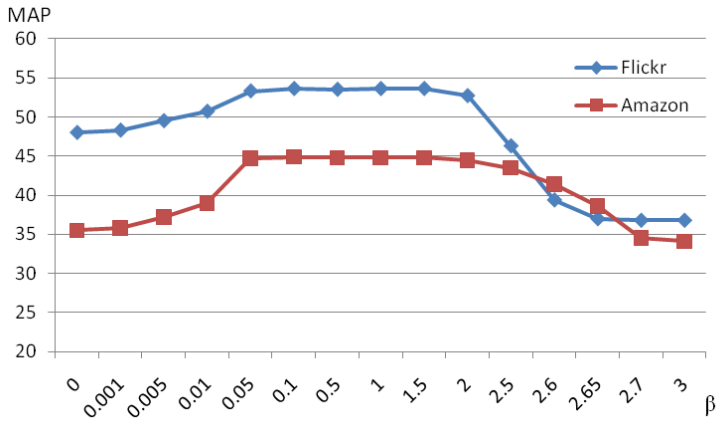


Figure 3.13: MAP w.r.t. parameter β. X-axis denotes β. Y-axis denotes MAP.

Convergence Figures 3.14 and 3.15 show $\Delta S = |S_{m+1} - S_m|$, the absolute change in the average sum of similarity scores (including images, groups and tags) from iteration m to $m + 1$ for algorithm IWSL_L on dataset Flickr and Amazon respectively. IWSL_G has similar results. We can see that IWSL converges very fast, and only 5 to 6 iterations are enough for most scenarios.

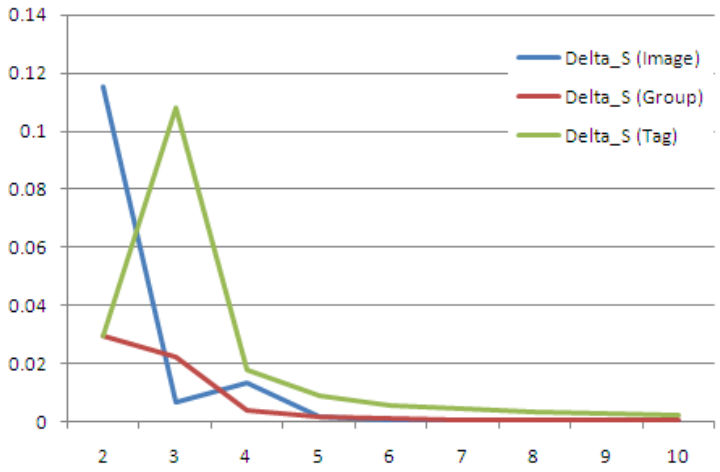


Figure 3.14: Convergence of IWSL on Flickr data. X-axis denotes the number of iteration. Y-axis denotes ΔS(i.e., *Delta_S*).

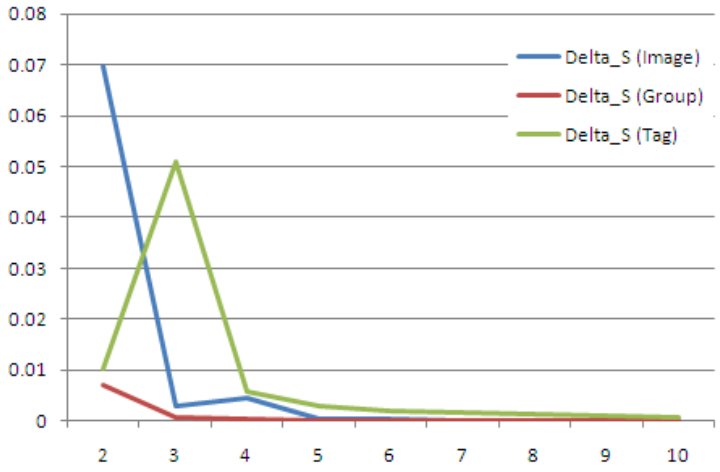


Figure 3.15: Convergence of IWSL on Amazon data. X-axis denotes the number of iteration. Y-axis denotes ΔS(i.e., *Delta_S*).

3.6 Application: Product Search and Recommendation for E-Commerce

Based on the proposed algorithm, a novel product recommendation system has been implemented for e-commerce to find both visually and semantically relevant products modeled in an image-rich information network. Figure 3.16 describes the system archi-

tecture. The bottom layer contains the product data warehouse which includes product images and related product information. The second layer performs meta-information extraction and image feature extraction. The third layer builds a weighted heterogeneous image-rich information network. The fourth layer performs information network analysis based ranking to find relevant results for a query. The top layer contains a user-friendly interface, which interacts with users, responds to their requests, and collects feedback.

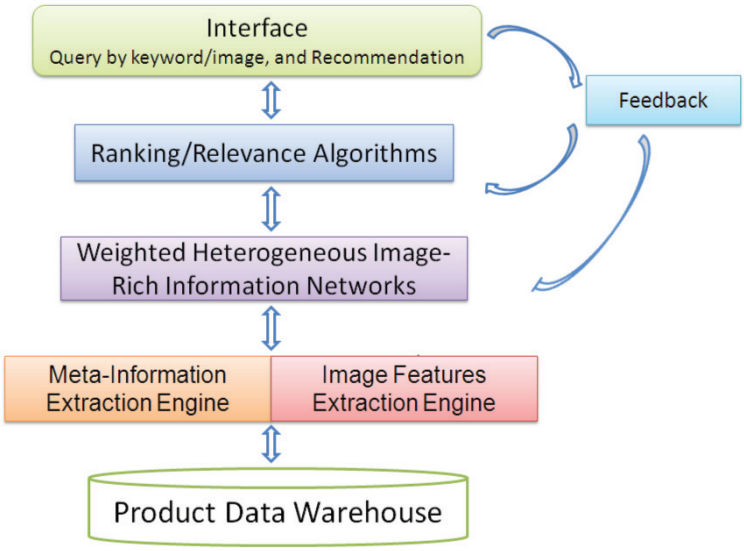


Figure 3.16: Product search and recommendation system architecture.

Figure 3.17 shows our product search and recommendation system for e-commerce using Amazon products as an example. When users search and click on a product in a web browser, we can recommend both visually and semantically relevant products, with the relevance score computed by the IWSL algorithm.

Figure 3.18 shows a comparison of our recommendation with the Amazon recommendation based on "Customers Who Bought This Item Also Bought" (which we call *co-bought*). When a consumer wants to buy a bag, our approach provides more relevant recommendations for the product. Figure 3.19 shows another comparison with the Amazon recommendation based on "Customers Who Bought This Item Also Viewed" (which we call *co-viewed*). The results are similar, because when a Amazon user wants to buy

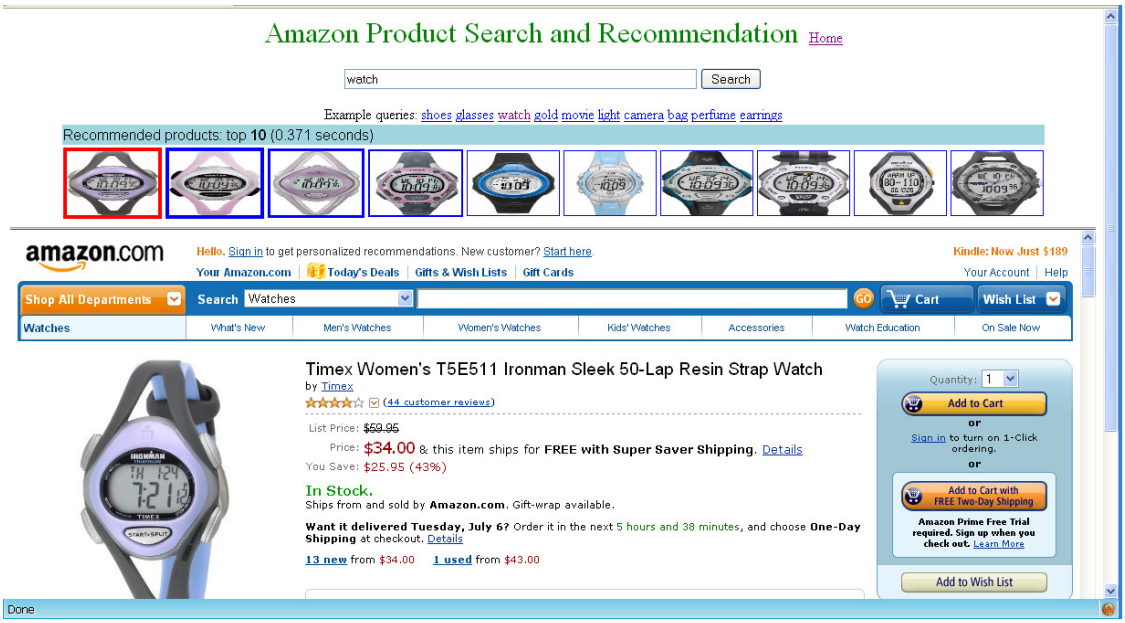


Figure 3.17: Snapshot of the product search and recommendation system for e-commerce.

a jewelry, he or she is likely to browse through similar products before making a final choice.

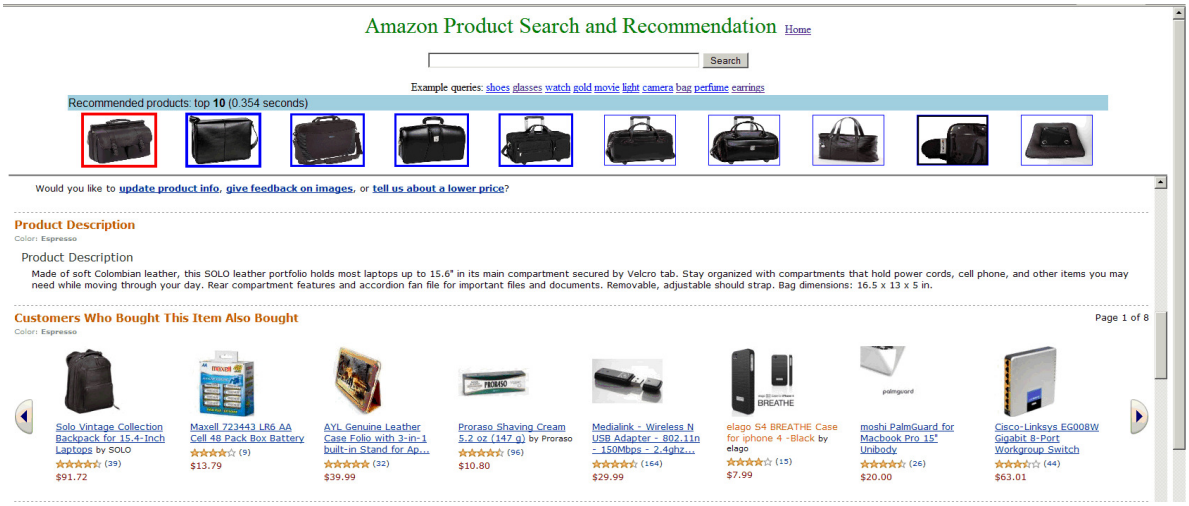


Figure 3.18: Recommendation comparison, ours v.s. Amazon's "Customers Who Bought This Item Also Bought."

One problem of using co-viewed or co-bought information for recommendation is that only the top-k ranking list is available, but the details (e.g., the frequency of such co-occurrence) are commercial secret and are not publicly accessible. Without such pro-

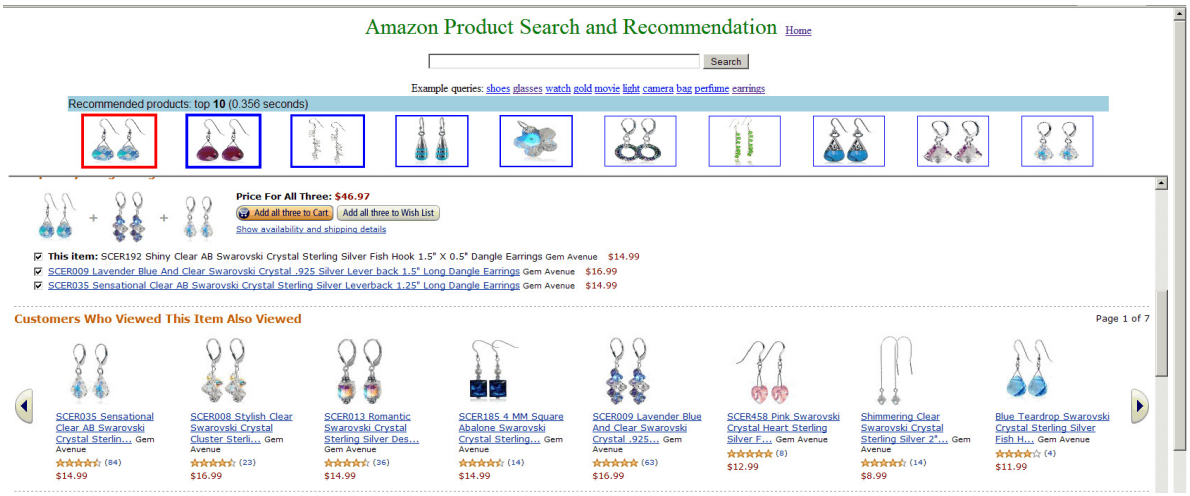


Figure 3.19: Recommendation comparison, ours v.s. Amazon's "Customers Who Bought This Item Also Viewed."

prietary information, it is difficult to combine sources from multiple e-commerce websites, such as Amazon, BestBuy, WalMart and Target, to generate an overall recommendation. We believe this is one of the reasons why general product search engines, such as Google Product Search and Bing Shopping, currently do not have such recommendation functions. However, for products (e.g., jewelry, watch, bag, glasses and clothes) that depend heavily on visual appearance to attract consumers, we are able to generate both visually and semantically relevant recommendations without the co-viewed or co-bought information. This leads to a better general product search service that enables users to compare and make the best choice.

Similar strategy can be applied to Flickr for photo and interest recommendation. When users browse Flickr photos, we can recommend relevant Flickr photos, or interest groups for users to join [126]. In addition, by integrating Flickr and Amazon networks, we can recommend relevant Amazon product photos to a Flickr photo for advertisement.

3.7 Conclusions

We present a novel and efficient way of finding similar objects (such as photos and products) by modeling major social sharing and e-commerce websites as image-rich information networks. Our major contributions are as follows:

(1) We propose HMok-SimRank to efficiently compute weighted link-based similarity in weighted heterogeneous image-rich information networks. The method is much faster than heterogeneous SimRank and K-SimRank.

(2) We propose both global and local feature learning approaches for learning a weighting vector to capture more important feature subspace to narrow the semantic gap.

(3) We propose the algorithm IWSL to provide a novel way of reinforcement style integrating with feature weighting learning for similarity/relevance computation in weighted heterogeneous image-rich information network.

(4) We conduct experiments on Flickr and Amazon networks. The results have shown that our algorithm achieves better performance than traditional approaches.

(5) We have implemented a new product search and recommendation system to find both visually similar and semantically relevant products based on our algorithm.

Chapter 4

Efficient Clustering of a Large Set of Content Objects

Given any similarity measure, data clustering can partition similar content objects into groups and provide a compact representation of the content relations. One big challenge for clustering content objects in social media is the large scale data. In this section, we propose GAD clustering framework based on activity detection as a general and fast clustering solution.

4.1 Overview

One of the big challenge of mining user content in social media is the large scale of data. Data clustering becomes a natural solution for this problem. Data clustering is one of the most popular data mining techniques with numerous applications. It has also been extensively studied in related research areas such as statistics, machine learning, pattern recognition, market research, biology, information retrieval and multimedia processing [40]. The most common scenario of clustering is as follows. Given a set of data objects, clustering groups the dataset into clusters, so that objects within the same cluster have high similarity between each other but are dissimilar to the objects in different clusters.

Researchers have proposed many clustering algorithms, such as Partitioning Clustering: K-Means [66] [70], K-Medoids (such as PAM [57], CLARA [57] and CLARANS [74]), EM clustering [25], Mean Shift [28], K-Way spectral clustering [105], etc.; Hierarchical Clustering: AGNES (Agglomerative), DIANA (Divisive), BIRCH [128], ROCK [39], Chameleon [56], etc.; Density-based (or Locality-based) Clustering: DBSCAN [30],

OPTICS [3], DENCLUE [41], etc.; Grid-based Clustering: STING [110], CLIQUE [1], WaveCluster [90], etc.; Ranking-based clustering: RankClus [96], NetClus [97], etc. and Graph/Network Clustering [32]: such as SCAN [118]. We focus on partitioning-based clustering which is widely used in many applications and can be also integrated with many other types of clustering algorithms.

Performing efficient clustering on large dataset is especially useful. There have been many papers published for fast clustering on large dataset. Some develop fast core clustering algorithms; whereas others develop pre-processing methods, such as sampling, subspace and compression, to reduce the dataset to a smaller size to achieve speedup. For example, CLARA [57] uses sampling strategies to reduce the size of data. BIRCH [128] compresses the original data using CF-tree and then employs the core clustering algorithm (e.g., K-Means) to perform the real clustering. In this study, we focus on developing fast core clustering algorithms.

K-Means [66] [70] is one of the most popular clustering algorithms, due to its high efficiency/effectiveness and wide implementation in many commercial/non-commercial softwares. In 2006 IEEE International Conference on Data Mining (ICDM'06), it was ranked the 2nd among top ten most influential data mining algorithms [115], just next to the classification algorithm C4.5. K-Means has been discovered by several researchers, including Lloyd (1957, 1982), Forgey (1965), Friedman and Rubin (1967), and McQueen (1967). The basic K-Means algorithm performs simple but effective clustering by iteratively partitioning a given dataset into K clusters.

The basic idea of K-Means is as follows. Given K initial cluster centers, assign each pattern/point to the nearest center, update the cluster centers by calculating the mean of the member patterns, and repeat the assignment-and-updating process until a convergence criterion is satisfied. Typical convergence criteria include the maximum number of iterations, difference on the value of the distortion function, etc.

For large-scale datasets, the major computation burden of K-Means clustering origi-

nates from the numerous distance calculations between the patterns and the centers [58]. To deal with the problem, fast algorithms with different strategies have been proposed, such as PDS [8], TIE [19], Elkan [9], MPS [86], PAN [79], DHSS [99], FAUPI [62], kd-tree K-Means [81], AKM [83], HKM [76], GT [58] and CGAUCD [63]. Those algorithms come from several research communities, such as data mining, machine learning, pattern recognition, multimedia processing and computer vision.

PDS (Partial Distortion Search) [8] cumulatively computes the distance between the pattern and a candidate center by summing up the differences in each dimension. The effectiveness of PDS depends on the quality of the current candidate, the number of dimensions and the order of dimension to cumulate (especially for dimension-skewed data). If the dimensionality is high, PDS may still needs to compute many dimensions to stop accumulation.

TIE (Triangular Inequality Elimination) [19] uses the triangle inequality condition for metric distance to prune candidate centers, thus reduces the number of distance calculations. TIE needs extra space $O(K^2)$ to save a distance matrix for the center vectors, and the entries are recalculated at the beginning of each partition.

Elkan (by Charles Elkan) [9] is an exact fast algorithm for metric distance by using some metric distance properties. It needs to save the distances between every two centers and re-compute them at each iteration. The algorithm only works for metric distance, and is not scalable to large K, the number of clusters, since it requires an additional $O(K^2)$ complexity in both space and time. Elkan is similar to TIE.

MPS (Mean-distance-ordered Partial Search) [86] is especially designed for Euclidean distance. An efficient implementation involves using sorting to initially guess the center whose mean value is the closest to that of the current point and prune candidates via an inequality based on an Euclidean distance property. MPS is faster than K-Means if the improvement gained from pruning exceeds the overhead caused by sorting.

PAN [79] rejects unlikely centers using mean values and variances of an input vec-

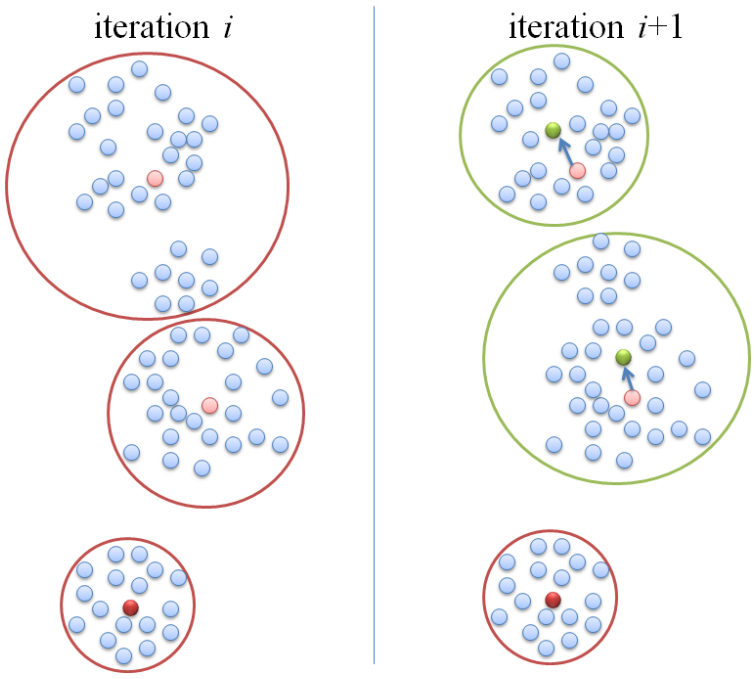


Figure 4.1: Active centers. In iteration i, there are three clusters, and the red points indicate the cluster centers. At the next iteration i+1, two light red points are active centers because they move to different locations, while the solid red point is static.

tor and its two sub-vectors. DHSS (Dynamic Hyperplanes Shrinking Search) [99] uses projection values of input vectors and centers on some dimensions to eliminate unlikely candidate centers. The DHSS algorithm with three projections has the less computing time than PAN. FAUPI [62] is another fast-searching algorithm using projection to reduce the dimension and inequality to reject unlikely codewords.

Many of the above algorithms depend on the metric properties and thus only works for metric distances. In this study, we explore another way - activity detection - which avoids the metric properties, thus works for both metric and non-metric distances. Activity detection is to identify and mainly focus on computation related to active centers, as shown in Figure 4.1.

Figures 4.2 and 4.3 show the percentage of active centers at each iteration with different number of clusters K. The vertical lines indicate the end of the iteration, there are different lines because different K may need different number of iterations to converge.

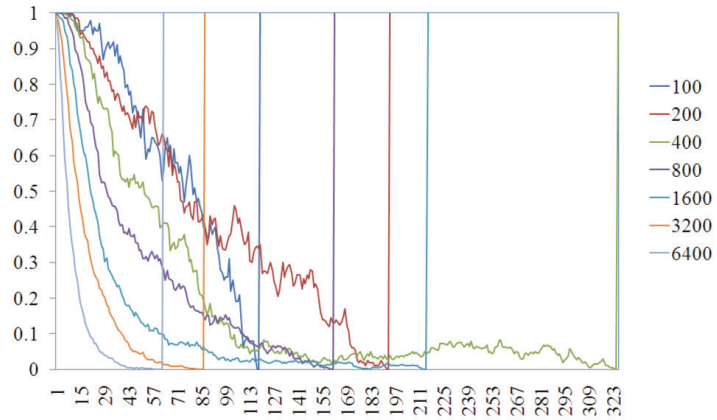


Figure 4.2: Activity percentage curves for different number of clusters, based on dataset VQDC. Horizontal axis denotes the number of iterations reached; vertical axis denotes the percentage of active centers at the specific iteration. Different lines represent different K, the number of clusters.

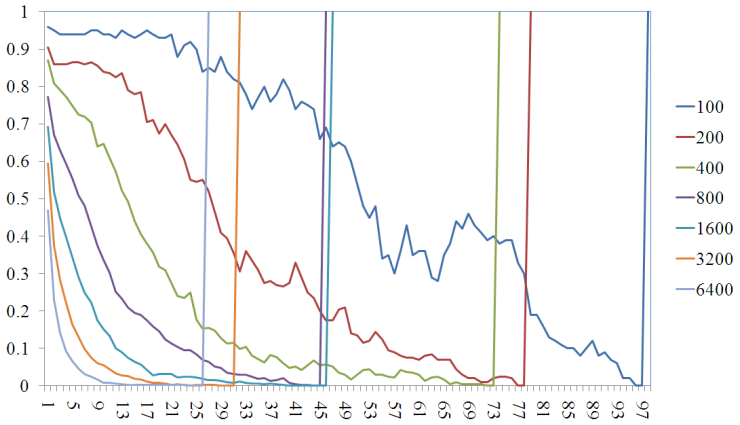


Figure 4.3: Activity percentage curves for different number of clusters, based on dataset HDS-MTI.

As shown in Figure 4.2 and 4.3, irrespective of the number of clusters, the percentage of active centers will generally decrease with the increase number of iterations; it means more and more centers are turning from active to static. **Active Area** is the area under the curve which contains the active centers. **Static Area** is the area above the curve which contains the static centers. This is the key aspect for activity detection to speed up clustering, because we can develop technique to focus on computing the active area and avoid the calculations associated with the static area.

Kaukoranta et al. proposed algorithm GT [58] to utilize point activity for fast clustering and showed that it can further speedup PDS [8], TIE [19] and MPS [86]. Lai et al. proposed algorithm CGAUCD [63] as an extension of GT and demonstrated that combining CGAUCD with MFAUPI (which is an extension of FAUPI [62]) achieves the highest speed.

GT and CGAUCD only partially explore the potential of activity detection, we will show in Section 4.5.3 that there actually exists a *lower-bound*. In this study, we propose a GAD (General Activity Detection) framework [47] to fully explore the power of activity detection for clustering. We design a set of algorithms (which are faster than GT and CGAUCD) within this framework for fast clustering in different scenarios. The most important contribution of our work is that GAD is the general solution to exploit activity detection for fast clustering and our algorithms within the framework can achieve very high speed.

4.2 General Activity Detection

In this section, we describe the notations, introduce the General Activity Detection framework (including definition and concepts), and then discuss the general idea of activity detection for improving clustering algorithm.

Let N be the number of patterns, D the number of dimensions, and K the number of centers. Suppose the algorithm runs I iterations to converge. At each iteration i (i = 1,..., I), for a pattern p (p = 1,..., N), we have:

NC(i, p, j) represents pattern p's jth nearest center. In real implementation, the value of NC(i, p, j) is the center's id.

D_NC(i, p, j) represents the distance from pattern p to its jth nearest center.

Dist(i, p, C_j) represents the distance between pattern p and center C_j.

4.2.1 Definition and Concepts

We formally define the GAD (General Activity Detection) framework as a function of four parameters:

$$GAD(S, A, m, B)$$

where S denotes Search Methods, A denotes Activity States, m denotes the number of Nearest Centers, and B denotes Boundary.

In the following we discuss the concepts used in the GAD framework.

We keep m **Nearest Centers** for each pattern, the information needed is: ids of the m nearest centers, and distances from the pattern to the m nearest centers.

A center could have different types of activity states. Let VC[prev] and VC[curr] be center C's feature vector in the previous and current iterations, respectively. Denote D as the distance between VC[prev] and VC[curr]. We define three types of cluster centers.

- If $D > 0$, the center is an **Active Center** for the current iteration.
- If $D = 0$, the center is a **Static Center**.
- For approximate algorithms, if $D < \varepsilon$, the center is an ε**-Approximate Static Center**, where ε is a predefined positive threshold; else if $D \geq \varepsilon$, the center is an Active Center.

We define search methods to find the nearest center(s) of a pattern as follows.

- **Full Search** means search from all the centers to find a pattern's m nearest centers.
- **Whole Full Search** means perform Full Search for all the patterns.
- **Partial Search**, or named **Active Search**, means search from active centers, which are usually a portion of the whole centers.
- m**-Search** means search from a pattern's previous m nearest centers.

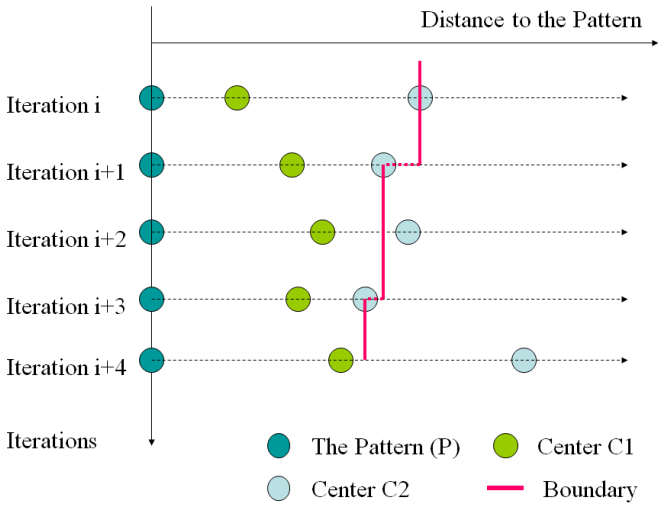


Figure 4.4: How Boundary changes. ($m = 2$)

- **0-Search**, a special case of m-Search, which just keeps the previous m nearest centers as the current m nearest centers, without doing any distance comparison.

We introduce the idea of Boundary to guarantee exact clustering result.

An **m-Boundary**, or simply named **Boundary**, is defined for each pattern. Whenever performing Full Search, the value of the Boundary is initialized as the distance from the pattern p to its mth nearest center. At any future iteration j, if the Boundary value is bigger than D_NC(j, p, m), the Boundary is updated to D_NC(j, p, m).

Property of the m-Boundary: One pattern's Boundary either shrinks or keeps unchanged, depending on how the new mth nearest center changes. The Boundary will not expand, except when Full Search is required and it is re-initialized to a value which is bigger than the current value.

Example 1. Figure 4.4 shows an example of how the Boundary of the pattern P changes at each iteration. To make the demonstration simpler, we assume the 3rd nearest center is static and only Partial Search is performed.

At iteration i, NC$(i, p, 1)$ = C1, NC$(i, p, 2)$ = C2, Boundary = D_NC$(i, p, 2)$.

At iteration i+1, C1 and C2 are active. NC$(i + 1, p, 1)$ = C1, NC$(i + 1, p, 2)$ = C2,

D_NC($i+1, p, 2$) < Boundary, update the Boundary to be D_NC($i+1, p, 2$).

At iteration i+2, C1 and C2 are active. NC($i+2, p, 1$) = C1, NC($i+2, p, 2$) = C2, D_NC($i+2, p, 2$) > Boundary, so the Boundary does not change. Boundary = D_NC($i+1, p, 2$).

At iteration i+3, C1 and C2 are active. NC($i+3, p, 1$) = C1, NC($i+3, p, 2$) = C2, D_NC($i+3, p, 2$) < Boundary, update the Boundary to be D_NC($i+3, p, 2$). □

4.2.2 Algorithm

We begin with analysis of GT and CGAUCD. GT is an exact clustering algorithm which is faster than K-Means and gets the same result. It saves each pattern's nearest center. The basic idea is that at each iteration, if a pattern's previous nearest center is static or moves closer to the pattern, search from active centers; otherwise, search from the full center set. CGAUCD extends GT by also considering each pattern's second nearest center.

The problem of GT and CGAUCD is the that they only consider the first nearest (and the second nearest) center, which is not able to fully explore the power of activity detection. We propose GAD (General Activity Detection) to consider any m number of nearest neighbors. Such extension is not a simple task, because we need to guarantee getting the exact result. To solve the problem, we introduce the idea of m-Boundary to make sure we can extend to any m without getting error. Our exact GAD algorithm is faster than GT and CGAUCD because it is able to reach a lower-bound (see details of the bound in Section 4.5.3) we found in activity detection. In addition, we introduce approximate GAD algorithms to further significantly improve the efficiency. Note that in Section 4.2.1 GAD introduces many new concepts which are studied in previous works. In the following sections, we describe our algorithms within the GAD framework in detail.

4.3 Exact GAD Algorithm

In this section, we present E-GAD (Exact GAD) algorithm within the GAD framework. E-GAD is a fast exact clustering algorithm which is faster than K-Means and GT while achieving exactly the same clustering result. The major concepts used in E-GAD include: Static and Active centers, Full Search and Partial Search, m-Boundary and m Nearest Centers.

The main goal of the E-GAD algorithm is to consider any m number of nearest neighbors with guaranteed ability to get the exact result by introducing the m-Boundary.

We describe the E-GAD algorithm procedure as follows.

Algorithm *E-GAD*

Input:

Data: N data patterns

Parameters: K, the number of centers; m, the number of nearest centers saved for a pattern.

Output: A set of K clusters and each pattern's m nearest centers.

Begin:

Step 1. Initialization (iteration $i = 1$).

1.1. Initialize K centers.

1.2. Mark all centers as Active.

1.3. Whole Full Search. For each pattern p ($p = 1, ..., N$), search for its nearest m centers from all candidate centers and initialize the Boundary as D-NC(i, p, m).

1.4. Update each center's activity state.

Step 2. Search method decision. For pattern p (beginning with $p = 1$) decide whether to perform Full Search or Partial Search. (iteration $i + 1$)

2.1. If its previous nearest center C_{prev1} = NC$(i, p, 1)$ is static at this iteration, perform Partial Search.

2.2. If NC$(i, p, 1)$ becomes active, calculate Dist$(i + 1, p, C_{prev1})$. There are three cases:

2.2.1. If the new distance Dist$(i + 1, p, C_{prev1})$ is smaller than the old distance D_NC$(i, p, 1)$ = Dist(i, p, C_{prev1}), perform Partial Search.

2.2.2. If Dist$(i + 1, p, C_{prev1})$ is bigger than D_NC$(i, p, 1)$, but smaller than or equal to the Boundary, perform Partial Search.

2.2.3. If Dist$(i + 1, p, C_{prev1})$ is bigger than both D_NC$(i, p, 1)$ and Boundary, perform Full Search.

Step 3. Update pattern p's nearest centers according to the search method decided by Step 2.

3.1. If Full Search is decided, search from all centers to find the m nearest centers, update the Boundary as D-NC$(i + 1, p, m)$.

3.2. If Partial Search is decided, there are three consecutive procedures to be done: (a) Check the pattern's previous m nearest centers and keep the static centers among them as candidates for the current m nearest centers; (b) find current m nearest centers from the candidates found in (a) and active centers; and (c) if D_NC$(i + 1, p, m)$ is smaller than the current Boundary, update the Boundary to be D_NC$(i + 1, p, m)$.

Step 4. Get next pattern $p = p + 1$. If $p < N$, go to Step 2; Else if $p = N$, go to Step 5.

Step 5. Assign each pattern to its nearest center. Calculate new center vectors and update the activity status of each center.

Step 6. Go to Step 2 until all the centers converge to stable status.

End

Example 3. Using Figure 4.5, we demonstrate how E-GAD works correctly for the example

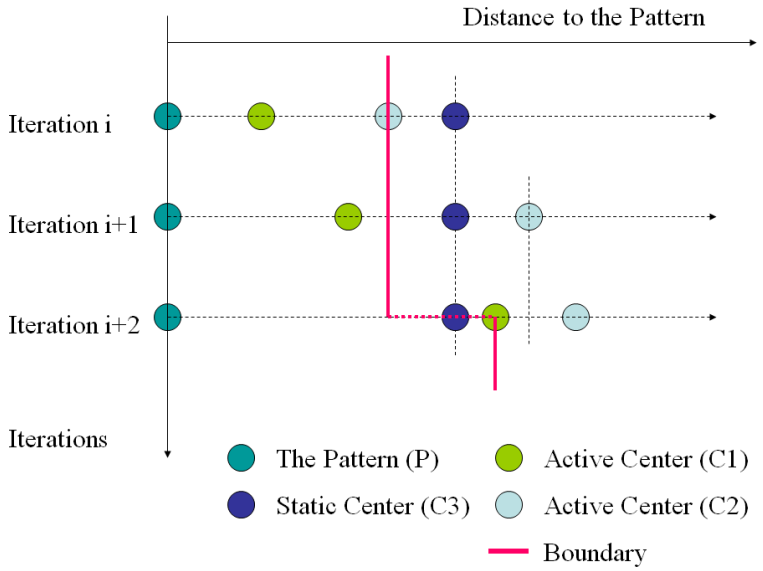


Figure 4.5: Illustrating how E-GAD works correctly for the case of Example 2.

case where CGAUCDB fails to get the right result. We ignore the fourth center since it is always the farthest to the pattern of interest. Take $m = 2$ as an example.

At iteration i, the value of the 2-boundary is Dist(i, p, C2).

At iteration $i + 1$, the new 2nd nearest center becomes farther and moves outside the boundary, so the boundary is unchanged and still Dist(i, p, C2).

At iteration $i + 2$, the new 1st nearest center moves out of the boundary, so we have to do Full Search for the pattern, both active and static centers will be explored. The new first nearest center will be C3, and the new second nearest center will be C1, which is the correct result. Since we have to do Full Search at iteration $i + 2$, the boundary will be re-initialized as the second nearest distance, i.e., Dist($i + 2, p$, C1). □

4.4 Approximate GAD Algorithms

This section presents Approximate GAD (AGAD) algorithms within the GAD framework, which can further accelerate the speed of E-GAD.

Based on different assumptions and different levels of approximation, we propose four AGAD algorithms.

Only the crucial parts are described for each algorithm, Steps 1, 4, 5 and 6 are similar to E-GAD.

4.4.1 NS-AGAD (Naive Static AGAD)

The assumption of NS-AGAD is that if a center is static at certain iteration, it will continue to be static at all future iterations. Thus we do not need to explore any other candidates. This is the most intuitive assumption. However, a static center in a certain iteration may become active in the future iterations.

We describe the major steps of the NS-AGAD algorithm as follows.

Algorithm *NS-AGAD*

Core Steps:

Step 2. Search method decision. (Full, Partial or 0-Search)

2.1. If pattern p's previous nearest center C_{prev1} is static at this iteration, perform 0-Search.

2.2. Same as E-GAD.

Step 3. Search and Update.

3.1 and 3.2. Same as E-GAD.

3.3. If 0-Search is decided, simply copy the previous m nearest centers as the new m nearest centers.

4.4.2 S-AGAD (Static AGAD)

S-AGAD is based on the assumption that if a pattern's former nearest center is static, the area near the pattern is relatively stable and the new nearest center will likely come from the pattern's previous m nearest centers, thus we avoid searching other centers.

We describe the major steps of the S-AGAD algorithm as follows.

Algorithm *S-AGAD*

Core Steps:

Step 2. Search method decision. (Full, Partial or m-Search)

2.1. If pattern p's previous nearest center C_{prev1} is static for this iteration, perform m-Search.

2.2. Same as E-GAD.

Step 3. Search and Update.

3.1 and 3.2. Same as E-GAD.

3.3. If m-Search is decided, search within previous m nearest centers and update the new order of nearest centers.

Different from NS-AGAD, in S-AGAD a pattern may need Full Search in some iterations even if it has static first nearest center in an earlier iteration.

4.4.3 I-AGAD (Inward AGAD)

The assumption of I-AGAD is that if a pattern's previous nearest center is static, or becomes active but moves inward to the pattern, the new nearest center is very likely to be that center or any other center from the previous m nearest centers, and we do not have

to search from other centers.

Compared with S-AGAD, I-AGAD also considers active centers, thus it will have stronger candidate center pruning ability.

We describe the core steps of the I-AGAD algorithm as follows.

Algorithm *I-AGAD*

Core Steps:

Step 2. Search method decision. (Full, Partial or m-Search)

2.1. If pattern p's previous nearest center C_{prev1} is static for this iteration, perform m-Search.

2.2. If center C_{prev1} is active, calculate Dist$(i+1, p, C_{prev1})$. There are three cases:

2.2.1. If Dist$(i+1, p, C_{prev1})$ is smaller than D-NC$(i, p, 1)$ (= Dist(i, p, C_{prev1})), perform m-Search.

2.2.2 and 2.2.3. Same as E-GAD.

Step 3. Search and Update.

3.1 and 3.2. Same as E-GAD.

3.3. If m-Search is decided, search within previous m nearest centers and update the new order.

4.4.4 WB-AGAD (Within-Boundary AGAD)

WB-AGAD is based on the assumption that no matter whether a pattern's previous nearest center is static or active at the next iteration, as long as it is still within the Boundary of the pattern, the new nearest center will likely be that center or some other center from the

Table 4.1: Characteristics of Approximate GAD algorithms

Algorithm	**DA**	***m*-Impact**	**Speedup**
NS-AGAD	less high	low	medium
S-AGAD	high	low	low
I-AGAD	high	medium	medium
WB-AGAD	medium	high	high
CGAUCDB	high	none	minus

previous m nearest centers, and thus avoid searching all other candidates.

Different from I-AGAD, WB-AGAD further relaxes the constraint to the Boundary, thus achieving more aggressive candidate pruning.

We describe the WB-AGAD algorithm as follows.

Algorithm *WB-AGAD*

The procedure of WB-AGAD can be generally described as replacing all the Partial Search of E-GAD by m-Search.

The above four approximate GAD algorithms, NS-AGAD, S-AGAD, I-AGAD and WB-AGAD, achieve different degrees of approximation. Table 4.1 lists the characteristics of the four algorithms and CGAUCDB. DA denotes the degree of approximation. m-Impact denotes the impact of the value of parameter m on the performance of the algorithm. Speedup is compared with E-GAD. They have different tradeoffs between clustering approximation and speed, the user may choose the one mostly meets the application requirement. These characteristics are supported by our experiment results.

Table 4.2: Basic algorithms within the GAD framework

Algorithms	Search Methods	Activity States	m	Boundary
K-Means	FS	A	0	No
GT	FS, PS	Ac, St	1	No
CGAUCDB	FS, PS	Ac, St	2	No
E-GAD	FS, PS	Ac, St	Any	Yes
NS-AGAD	FS, PS, 0S	Ac, St, (εS)	Any	Yes
S-AGAD	FS, PS, mS	Ac, St, (εS)	Any	Yes
I-AGAD	FS, PS, mS	Ac, St, (εS)	Any	Yes
WB-AGAD	FS, mS	Ac, St, (εS)	Any	Yes

4.5 Analysis of GAD

In this section we first analyze the GAD framework and then discuss the space and time complexity of the GAD algorithms.

4.5.1 Analysis of the GAD Framework

Based on our previous given formal definition of the GAD framework, Table 4.2 presents characteristics and parameters of the basic algorithms within the framework. FS (Full Search), PS (Partial Search), 0S (0-Search), mS (m-Search), Ac (Active), St (Static) and εS (ε-Approximate Static).

GT is a special case of E-GAD with $m = 1$ (with or without using Boundary gives the same result when $m = 1$). CGAUCDB is a special case of approximate GAD with $m = 2$ and without using Boundary.

Why GAD is the general solution for activity detection? We take the basic exact GAD algorithm E-GAD as an example to answer this question. Figures 4.6 and 4.7 show the percentage of Full Search patterns at each iteration, with different m. The area under a curve contains the Full Search patterns (we call **Full Search Area**).

If the Full Search Area is very small, the number of patterns which perform Full Search will be very small. Compared with other search methods, Full Search is the worst case,

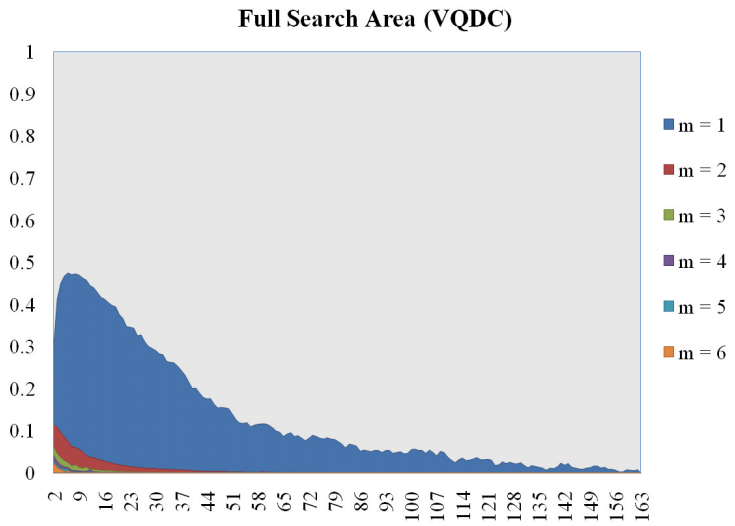


Figure 4.6: Full Search area (shown in different colors). Horizontal axis denotes the number of iterations reached; vertical axis denotes the percentage of Full Search points at the iteration. This result is based on dataset VQDC.

because it requires calculating the distances between the pattern and all centers (both active and static centers), which is very time consuming for clustering large scale data with many clusters.

Ideally we want no pattern to perform Full Search, that is to say, to make the Full Search Area be 0 (in this case, we only need to search the active area). This is exactly what GAD can provide. With the increase of m, the area will be smaller and smaller. The reason is that bigger m makes the m-Boundary bigger, thus keeping more candidate centers within the boundary and avoiding more Full Search.

As shown in Figure 4.6 with the VQDC dataset (described in section 4.6.1), when $m = 0$, Full Search is performed for all patterns, the Full Search Area is the whole rectangle area. When $m = 3$, the area becomes very small, very few patterns need to do Full Search. When $m = 6$, the area is most 0, thus almost no pattern needs to perform Full Search. We see similar phenomena in another example using the HDS-MTI dataset (described in section 4.6.1), as shown in Figure 4.7.

We can see that GAD is able to make Full Search Area as small as possible. Since

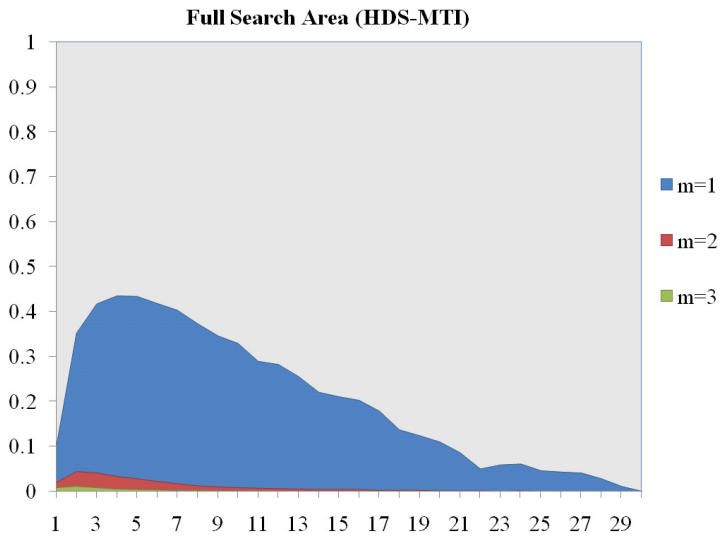


Figure 4.7: Full Search area (shown in different colors). Horizontal axis denotes the number of iterations reached; vertical axis denotes the percentage of Full Search points at the iteration. This result is based on dataset HDS-MTI.

decision on whether to perform Full Search is based on the activity of the centers, GAD is the general solution to exploit activity detection for fast clustering.

4.5.2 Space Complexity

The space complexity of GAD algorithms is $O(N(D + m) + KD)$. For K-Means, $m = 0$. For E-GAD and AGAD algorithms, besides saving N patterns and K centers, we also save a Boundary, the indexes and distances of the pattern's m nearest centers for each pattern. However, m is usually a small number. For E-GAD, we usually only need m equal to 3 to achieve the almost optimal result; for AGAD algorithms, m is also relatively small. For large scale data, N is often much bigger than D and m; moreover, in high dimension case, $D \geq m$. So we could claim that the GAD algorithms have almost the same level of space complexity.

4.5.3 Time Complexity

We analyze the time complexity of GAD by the number of distance calculations needed. In general, the time complexity is $N * f(K, I)$.

$$f(K, I) = \sum_{i=1}^{I} (P_{full}(i) * K + (1 - P_{full}(i)) * N_{active}(i))$$

where $P_{full}(i)$ is the percentage of Full Search patterns at iteration i, $1 - P_{full}(i)$ is the percentage of Partial Search patterns at iteration i, and $N_{active}(i)$ is the number of active centers at iteration i.

For K-Means, $P_{full}(i) = 1$, so $f(K, I) = K * I$, which is the whole space including both the static area and the active area.

GAD is able to make $P_{full}(i)$ close to 0, and thus $f(K, I) = \sum_{i=1}^{I} N_{active}(i)$, which is the active area, which is only a portion of the whole area $K * I$. This is the reason why E-GAD can be much faster than K-Means. The complexity of an exact algorithm is **lower-bounded** by the active area and GAD is able to achieve this bound.

For approximate GAD algorithms, we take WB-AGAD as an example to analyze. It uses m-Search instead of Partial Search, thus avoids a lot of active centers, and $f(K, I)$ becomes much smaller than the active area. This is the reason why WB-AGAD can be much faster than E-GAD.

In large scale and large clusters clustering case, K is very big, fast algorithms like (TIE) [19] and Elkan [9] becomes very slow since they need an additional $K * K * I$ computations.

4.5.4 Extended GAD Algorithms

In this section, we extend the basic GAD algorithms to consider the following two problems. How to use GAD for very large number of clusters? How to improve the quality of the clustering result?

GAD for Very Large Clusters

Most existing fast clustering algorithms only work for small or medium cluster size K. However, many large scale applications expect very large K, such as large scale web image clustering, codebook generation and vector quantization.

HKM [76], kd-tree K-Means [81] are among the fastest (but with a decrease in clustering quality) algorithms which work for the *large clusters* problem because their time complexity on K is only $O(log(K))$. We discuss how to use our GAD framework to get even faster performance and achieve improved clustering quality.

H-GAD (Hierarchical GAD)

We propose Hierarchical GAD (H-GAD) to perform hierarchical clustering in the way similar to HKM [76], but uses GAD as the core clustering algorithm. The basic idea is as follows. An initial GAD process (any of the basic GAD algorithms can be used) runs on the root node which contains the whole patterns and partitions them into k clusters, each cluster as a child node. H-GAD recursively applies the process to each node. The clustering tree is built level by level, and stops when reaching the maximum level L. There are k^L final clusters at the bottom level of the resulting tree.

One major problem of the clustering tree is that it minimizes the distortion functions for k clusters locally at individual nodes which contain only part of the data patterns (except for the root node) and thus cannot achieve the global minimization optimized by clustering directly for k^L clusters. To get the same number of final clusters, bigger L makes k smaller and thus makes the whole process faster. However, with the increase of L, it will be more local to small nodes and the performance will drop accordingly.

H-GAD is better than HKM because of the following two reasons:

- GAD is faster than K-Means, so H-GAD can be faster than HKM when performing clustering at each node.
- When perform clustering for the same clusters, H-GAD can finish the whole com-

putation at the same time as HKM but with bigger k and smaller L, thus improves the performance.

In reality, H-GAD can even achieve better clustering quality than HKM with less running time.

KD-GAD (kd-tree GAD)

kd-tree [12] has been used to perform approximate nearest neighbor search to speed up clustering [81]. AKM [83] uses random kd-tree forest to make kd-tree more robust. However, this robust improvement can be simply achieved by a standard kd-tree using a larger number of leaf nodes for exploration. Our preliminary experiment shows that kd-tree works better than random kd-tree forest.

kd-tree (or other fast approximate nearest neighbor search algorithms, such as LSH [23]) can be integrated with the GAD framework (we call KD-GAD). Take E-GAD as an example, we can build two kd-trees, one for all centers which we call the *Full kd-tree*, another for active centers which we call the *Active kd-tree*. When perform Full Search in E-GAD, instead of searching all the centers, we search from the Full kd-tree; similarly, when perform Partial Search, we search from the Active kd-tree.

Naively combining kd-tree and E-GAD using Static Center gets slightly better clustering quality but decreases the speed, because kd-tree makes most centers active. So we use ε-Approximate Static Center in stead of Static Center to make more and more centers become static, approximately.

kd-tree E-GAD can be faster and obtains better clustering quality than kd-tree K-Means. The reason being faster is because we can use Partial Search and converge sooner. The reason for better quality is because it can keep the current nearest centers, and avoid missing them in future iterations of kd-tree search.

Clustering Quality Improvement

Error Accumulation Effect is an inherent problem for many approximate iteration algorithms. We propose Regular Whole Full Search (RWFS) to partially solve the problem and improve the clustering quality.

The basic idea of RWFS is to perform Whole Full Search (WFS) regularly to find the true nearest centers, thus eliminate the errors. We use a factor R to control when to perform WFS again. There could be different ways to decide the factor R. One simple method is setting it as a constant value and after R iterations we perform WFS no matter how many centers are already static.

GAD can also work with many existing clustering quality enhancing techniques, such as Swapping [55], Bagging [29], Boosting [34] and Clustering Ensemble [94] (or Consensus Clustering [5]). We discuss two examples:

1. *GAD with Swapping*. We can perform the swapping operations before updating the center activity state in GAD. Because swapping makes a larger percentage of centers active, and some of them just change slightly, using ε-Approximate Static Center is recommended to achieve fast convergence.

2. *GAD with Clustering Ensemble*. Since GAD can perform very fast clustering, we may generate multiple clustering results by using different initialization schemes or different feature subspaces, and then use clustering ensemble technique to get a final result which is better than using only a single clustering.

4.6 Experiments

We present extensive experimental evaluation for the GAD algorithms in this section. Experiments were conducted on a PC with a 3.4GHz Pentium D CPU and 1GB RAM.

To evaluate time performance, we use the Speedup of algorithm A over baseline B:

$$Speedup(A, B) = \frac{T_B}{T_A}$$

where T_A is the execution time of A and T_B is the execution time of B. If $Speedup(A, B) > 1$, algorithm A is faster than the baseline B.

To evaluate clustering quality, we use Sum of Distance ratio (SDR):

$$SDR(A, B) = \frac{SD(B)}{SD(A)} \times 100\%$$

where SD is the sum of distances between each pattern and its center. Squared Euclidean distance is used in our experiments. If $SDR > 1$, algorithm A gets better clustering quality than the baseline B. Note that activity detection is not based on the metric property of metric distances, such as Euclidean and L1. So if the distance used is the more general formulation of any Bregman divergence [7], then the clustering quality measure will be the ratio of Bregman losses.

4.6.1 Datasets

Vector Quantization is a classical signal processing technique and is used in areas such as data compression and density estimation. GAD can be used for vector quantization based data compression. We generated a dataset from six standard gray images: Baboon, Boats, Bridge, Couple, Goldhill, and Lena. 4×4 spatial pixel blocks were constructed for each image and each block is represented by the pixel value. The more number of clusters are, the better the quality of compression and the less compression rate we can obtain. We call this dataset **VQDC**.

Large Scale Image Clustering can help implement efficient images retrieval systems and create a user-friendly interface to the large image database. We test our algorithms for

Table 4.3: Statistics for datasets used in experiments

Datasets	Samples	Dimensions	Application
VQDC	117,376	16	Data Compression
KDDCUP04Bio	145,751	74	Bioinformatics
HDS-MTI	30,000	1024	Very High Dimension Image Clustering
MTI	1,608,325	100	Very Large Scale Image Clustering

this application, using the dataset collected by Torralba et al. [103]. They gathered from the web 79 million images using queries of 75,062 non-abstract English nouns listed in the Wordnet and provide a subset of about 1.6 million images which were arranged by about 53,000 query words [102]. We convert the images to be 10×10 grey format and use the grey levels as features. We call this dataset **MTI**.

A subset of MTI images with $32 \times 32 = 1024$ grey features is used to test the performance of the clustering algorithms in very high dimension. We call this dataset **HDS-MTI**.

The fourth dataset we use is from the protein homology prediction task of KDDCUP 2004 [59]. There are 75 features which describe the sequence alignment match between the native protein sequence and the sequence that is tested for homology. We call this dataset **KDDCUP04Bio**.

Table 4.3 summarizes the datasets. The number of dimensions varies from tens to thousands, and the number of samples varies from tens of thousands to over one million.

Figures 4.8, 4.9, 4.11 and 4.10 show the datasets projected to their first three principle components based on PCA analysis [53]. Note that the datasets are in high dimensions, and the projection on the first three principle components only reflect part of the information and cannot be used to claim the real clusters on the original high dimensions.

4.6.2 Performance Result

Performance of Exact GAD In this section, we analyze the impact of m on E-GAD and compare E-GAD with other exact algorithms.

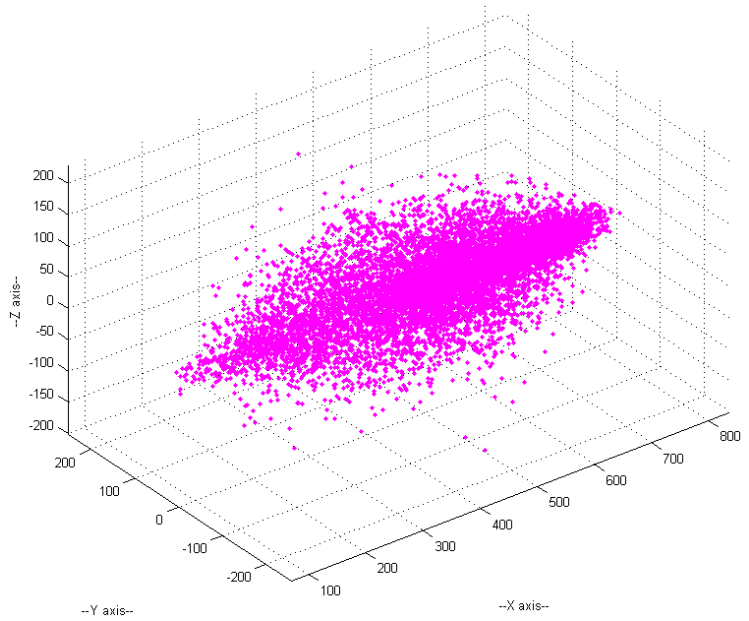


Figure 4.8: Projection on the first three principle components for dataset VQDC.

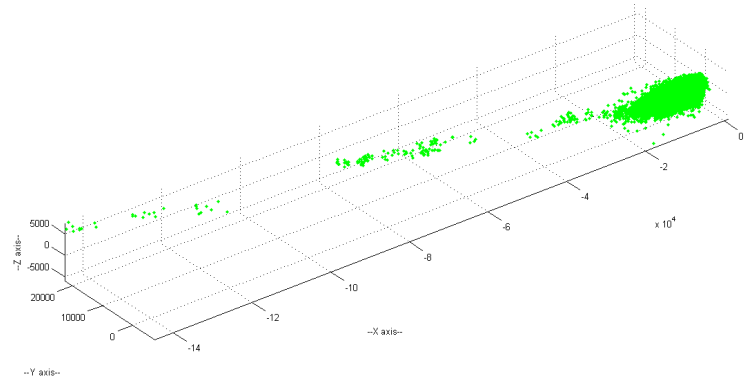


Figure 4.9: Projection on the first three principle components for dataset KDDCUP04Bio. There are some outliers, but most points lay on the right side.

Impact of m on E-GAD

Figures 4.12, 4.13 and 4.14 report how m impacts the Speedup of E-GAD over the baseline algorithm K-Means for datasets VQDC, HDS-MTI and KDDCUP04Bio. The cluster number is 2000.

From the results we can see that the best time performance is usually achieved at m being 3 or 4. When continue to increase m, the improvement on the speed is limited, because the increase on the size of Full Search Area is small. With too big m, the speed

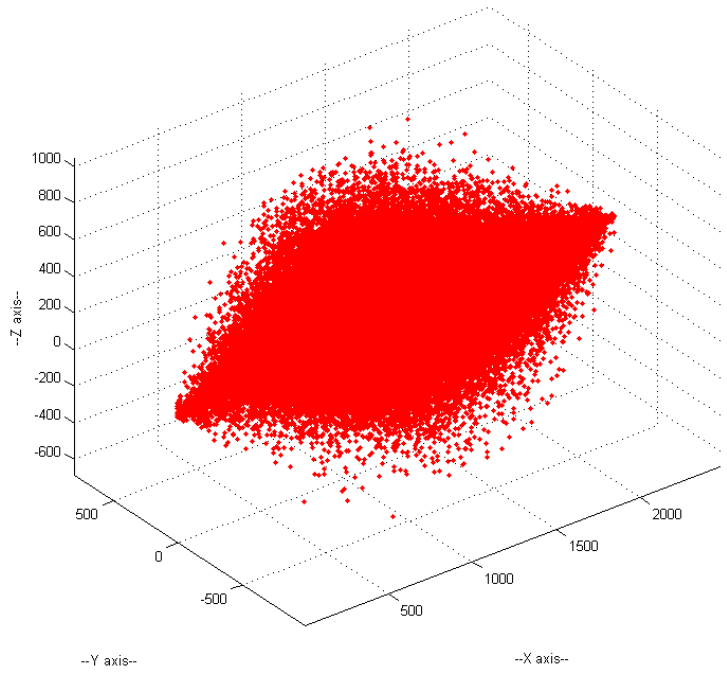


Figure 4.10: Projection on the first three principle components for dataset MTI.

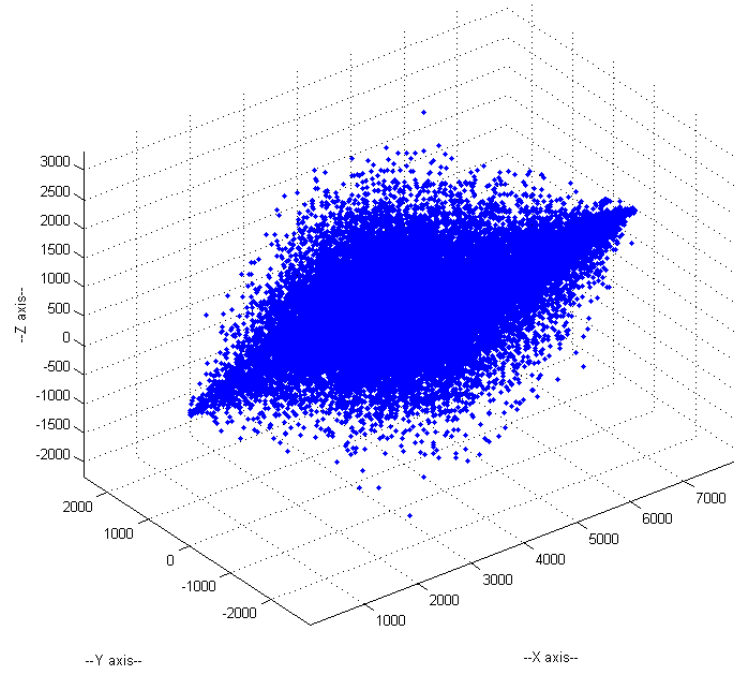


Figure 4.11: Projection on the first three principle components for dataset HDS-MTI.

slightly decreases, because it takes more time to keep the larger number of nearest centers sorted. In general, we can simply choose $m = 3$ for E-GAD.

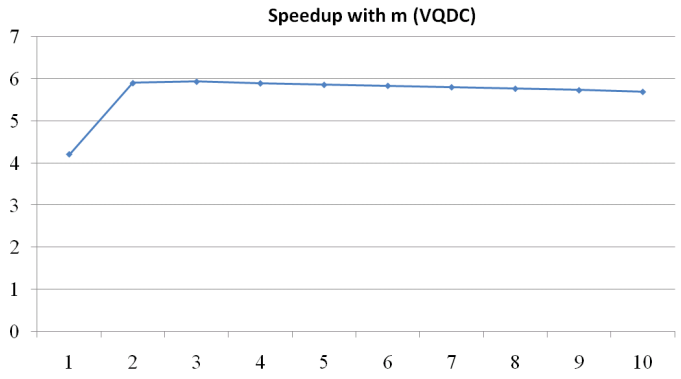


Figure 4.12: Impact of parameter m on E-GAD for dataset VQDC. The horizontal axis denotes the value of m; the vertical axis denotes the Speedup over K-Means.

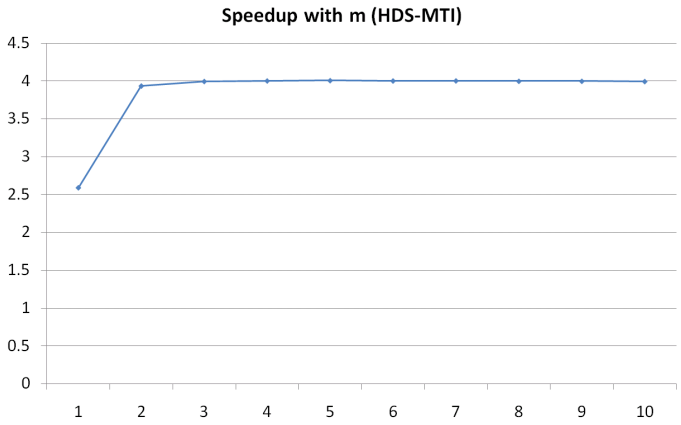


Figure 4.13: Impact of parameter m on E-GAD for dataset HDS-MTI. The horizontal axis denotes the value of m; the vertical axis denotes the Speedup over K-Means.

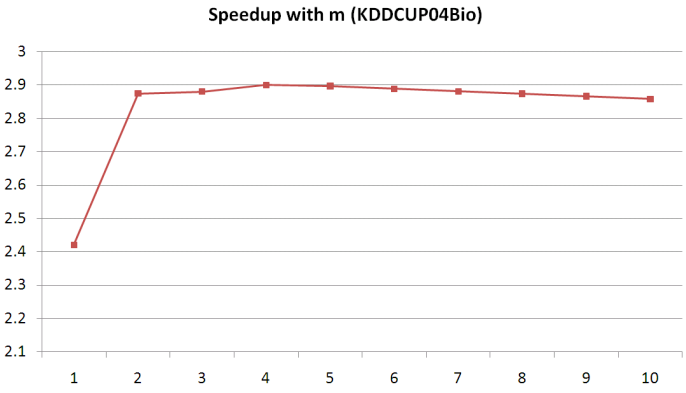


Figure 4.14: Impact of parameter m on E-GAD for dataset KDDCUP04Bio. The horizontal axis denotes the value of m; the vertical axis denotes the Speedup over K-Means.

Compare Exact Algorithms

We compare E-GAD with K-Means and GT. The goal is to evaluate the speed of GAD when we want to get exactly the same clustering result. Figures 4.15, 4.16 and 4.17 show the Speedup of E-GAD (with $m = 3$) and GT over the baseline algorithm K-Means on datasets VQDC, KDDCUP04Bio and HDS-MTI, respectively. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over K-Means. Since K-Means is the baseline algorithm, its value is always 1.

The performance of E-GAD is always the best. E-GAD is generally several times faster than K-Means. The highest speedup is observed in very high dimension dataset HDS-MTI where E-GAD is over 10 times faster than K-Means when the number of clusters is 6400.

The general trend is that the larger the number of clusters, the faster E-GAD could be. The reason is that as the number of clusters becomes larger, the computation burden on searching for the nearest center of each point becomes more obvious; the advantage of E-GAD also becomes more obvious, since E-GAD can avoid many computations and mainly focus on the Full Search Area. Another reason is that the speedup of E-GAD depends on the percentage of active centers, larger cluster number may leads to higher percentage of active centers. However, the percentage may also be impacted by other factors, such as center initialization, data dimensions, data type and the inherit data clustering structure. For a larger number of clusters, it's possible to see a slight drop in Speedup if the other factors make a negative contribution to the percentage. This seasons may explain why there exists zigzag in the curves.

Performance of Approximate GAD Algorithms In this section we analyze the impact of m on Approximate GAD algorithms and compare the performance of relevant algorithms.

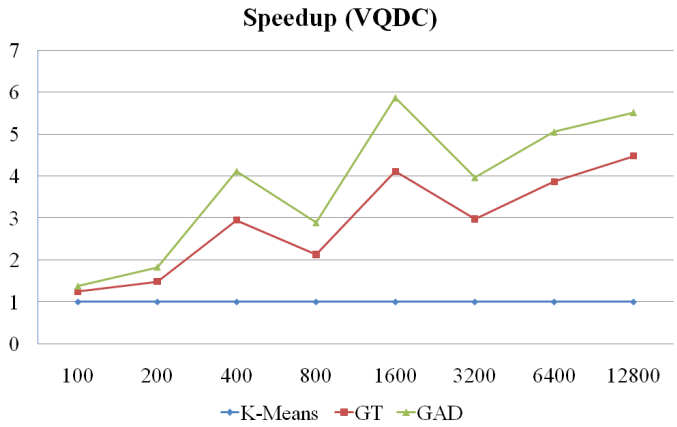


Figure 4.15: Time performance of E-GAD, K-Means and GT on dataset VQDC. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over K-Means.

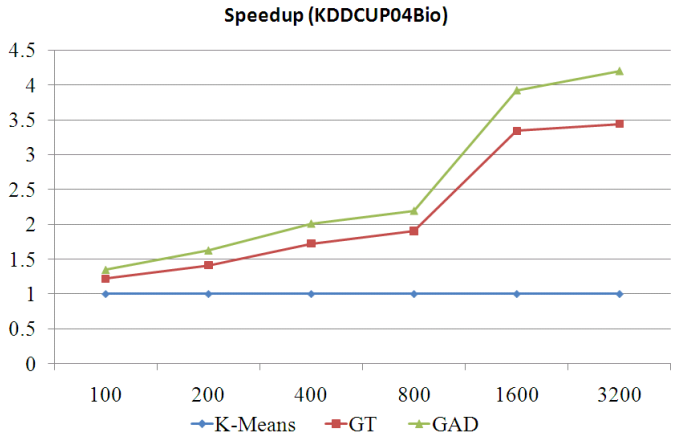


Figure 4.16: Time performance of E-GAD, K-Means and GT on dataset KDDCUP04Bio. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over K-Means.

Impact of m

Figure 4.18 shows the impact of parameter m on the clustering quality of the four approximate GAD algorithms: NS-AGAD, S-AGAD, I-AGAG and WB-AGAD. The number of clusters is 1000.

The results show that NS-AGAD, S-AGAD and I-AGAG can achieve high clustering quality even when m is very small. S-AGAD and I-AGAD have the best clustering quality; sometimes they are even slightly better than the exact result. I-AGAD has better Speedup

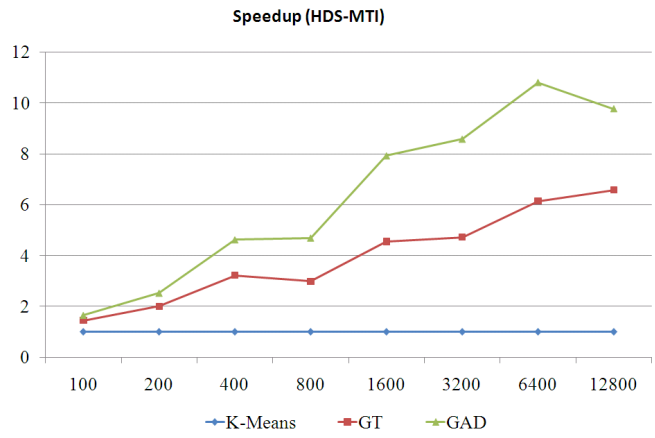


Figure 4.17: Time performance of E-GAD, K-Means and GT on dataset HDS-MTI. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over K-Means.

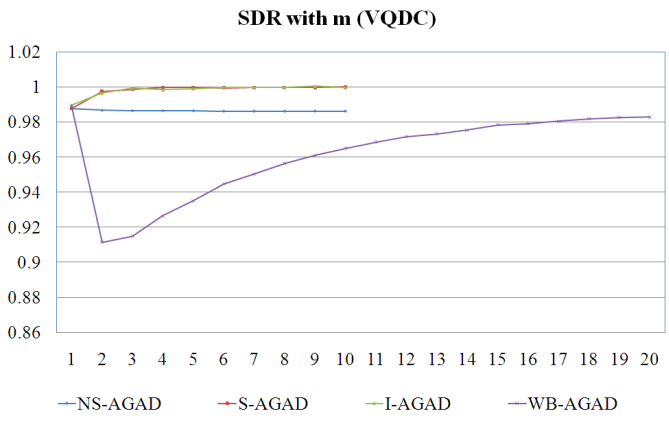
than NS-AGAD and S-AGAD. NS-AGAD is better than S-AGAD in Speedup but worse in SDR.

WB-AGAD is more impacted by m, because it uses m-Search instead of Partial Search. As we have mentioned before, when $m = 1$, WB-AGAD is identical to I-AGAD, so the SDR is same as I-AGAD at this point. When $m > 1$, larger m makes WB-AGAD get better result, it is because a center is less likely to move outside of the Boundary if the m is large. However, when m is small, the Speedup of WB-AGAD is most significant. Too large m takes more time to sort the m nearest centers.

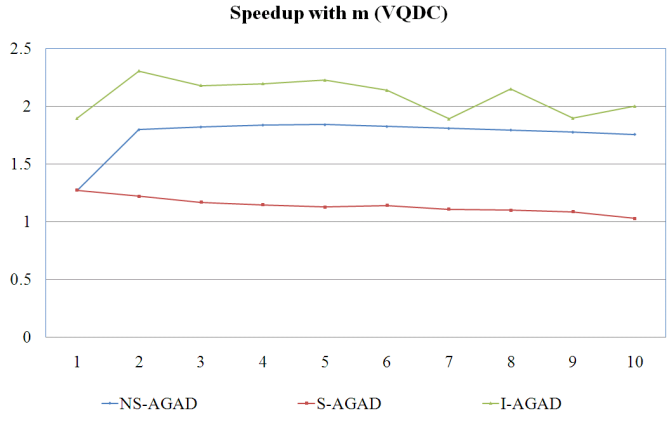
In most cases, setting $m = 5$ for NS-AGAD, S-AGAD, I-AGAG and $m = 15$ for WB-AGAD can make sure they achieve clustering quality of SDR higher than about 98% compared to the baseline exact result of E-GAD.

Performance Comparison

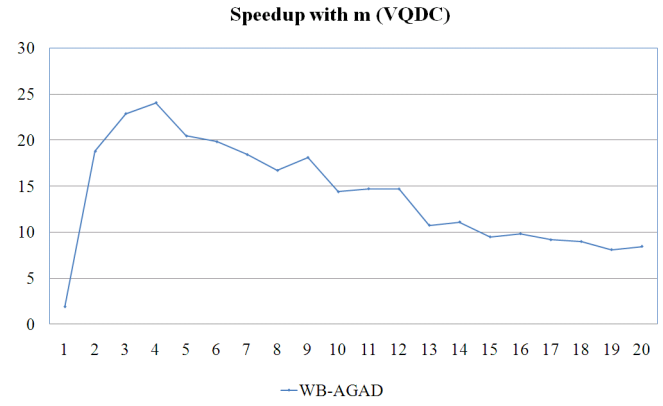
We perform experiments to compare the performance of four approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG WB-AGAD) and CGAUCDB on several datasets. Speedup and SDR are calculated over the baseline E-GAD, which we have already demonstrated to be the fastest exact algorithm compared with K-Means and GT. We set $m = 5$



(a)



(b)



(c)

Figure 4.18: Impact of parameter m on approximate GAD algorithms. Horizontal axis denotes the value of m; vertical axis denotes the SDR or Speedup over E-GAD. The curves are based on dataset VQDC; other datasets have generally similar results.

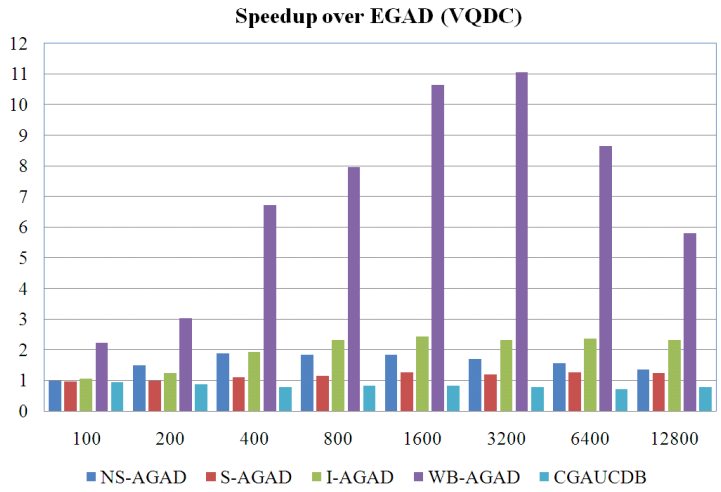


Figure 4.19: Comparison of approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG and WB-AGAD) and CGAUCDB for dataset VQDC. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over E-GAD.

for NS-AGAD, S-AGAD, I-AGAG and $m = 15$ for WB-AGAD.

Figures 4.19, 4.20 and 4.21 show the results on Speedup in datasets VQDC, KDCUP04Bio and HDS-MTI respectively. All the four approximate algorithms are faster than E-GAD, while CGAUCDB is always slower than E-GAD, i.e., its Speedup is always less than 1. I-AGAD can achieve very high clustering quality (bigger than 99%) and a Speedup mostly over 2. WB-AGAD achieves very high Speedup in most cases and gets clustering quality mostly within around 98%. In general, WB-AGAD can be around 10 times faster than E-GAD. The best performance we observed is at dataset KDDCUP04Bio where WB-AGAD is over 20 times faster than E-GAD when clustering 400 clusters.

In most cases, the approximate GAD algorithms can achieve good clustering quality. Figures 4.22 and 4.23 show the SDR scores in datasets VQDC and HDS-MTI, respectively. Note that VQDC is in low dimension, HDS-MTI is in high dimension.

Performance of GAD for Very Large Clusters Tables 4.4 and 4.5 report the performance of GAD algorithms H-GAD and KD-GAD for very large clusters on datasets VQDC and MTI. For dataset VQDC, we perform 10,000 clusters (to achieve a compression rate of 12). For dataset MTI, there are about 1.6 million images arranged by about 53,000 query

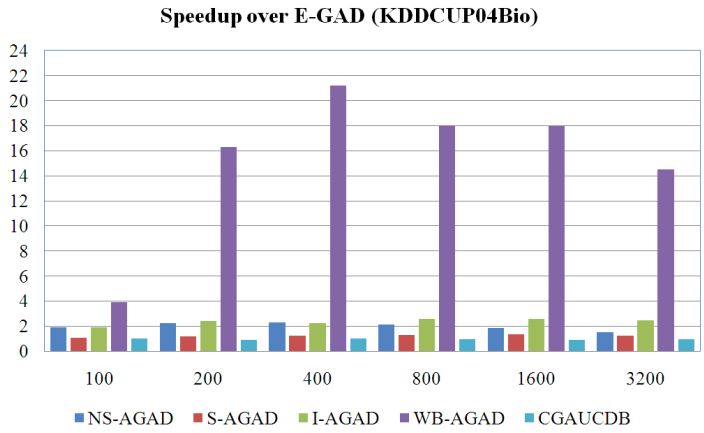


Figure 4.20: Comparison of approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG and WB-AGAD) and CGAUCDB for dataset KDCUP04Bio. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over E-GAD.

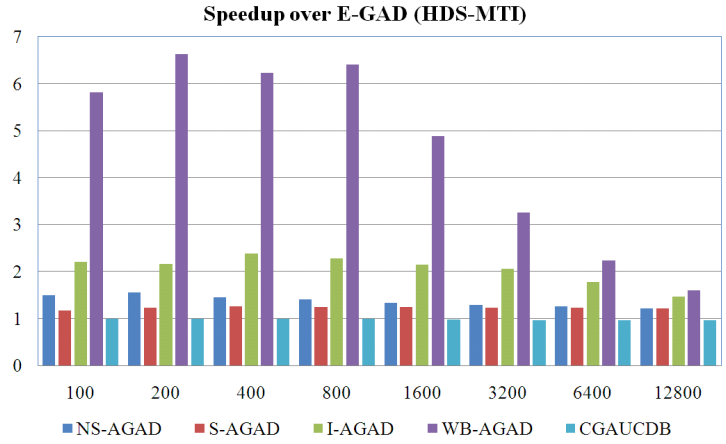


Figure 4.21: Comparison of approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG and WB-AGAD) and CGAUCDB for dataset HDS-MTI. Horizontal axis denotes the number of clusters; vertical axis denotes the Speedup over E-GAD.

words, and we do clustering of 50,000 clusters.

H-GAD performs hierarchical GAD clustering like HKM, and KD-GAD performs kd-tree based clustering. So we compare H-GAD with HKM, KD-GAD with kd-tree K-Means. For H-GAD, WB-AGAD (m = 10) is used as the basic clustering algorithm; for KD-GAD, E-GAD is used as the core clustering algorithm. The result shows that both H-GAD and KD-GAD can be faster than their counterpart algorithm while even getting better

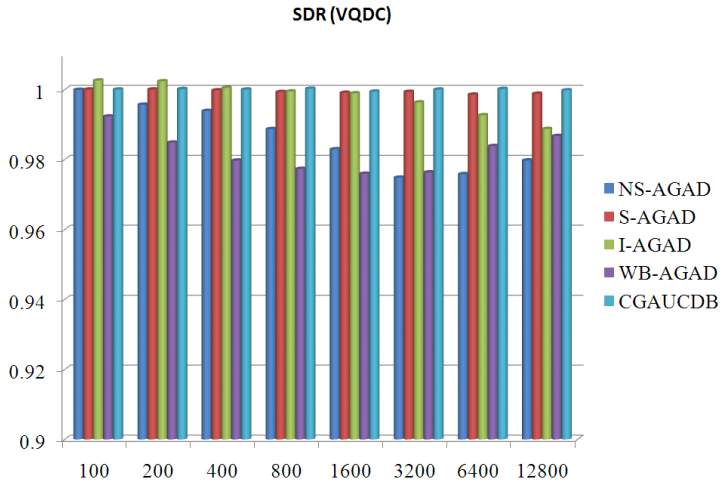


Figure 4.22: Clustering quality comparison of approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG and WB-AGAD) and CGAUCDB for dataset VQDC. Horizontal axis denotes the number of clusters; vertical axis denotes the SDR over E-GAD.

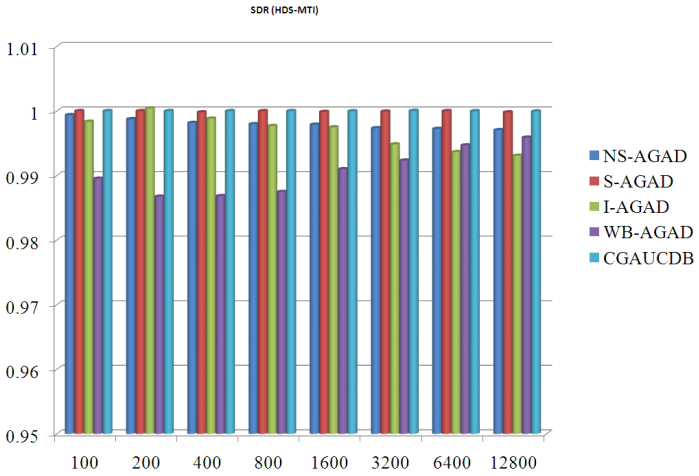


Figure 4.23: Clustering quality comparison of approximate GAD algorithms (NS-AGAD, S-AGAD, I-AGAG and WB-AGAD) and CGAUCDB for dataset HDS-MTI. Horizontal axis denotes the number of clusters; vertical axis denotes the SDR over E-GAD.

clustering quality.

Performance of RWFS We present the performance evaluation for the clustering quality improvement method RWFS, using SDR as the measure with WB-AGAD ($m = 5$) as the basic clustering algorithm and baseline. Since the factor R is the parameter which

Table 4.4: Performance of H-GAD compared with HKM

Datasets	SpeedUp	SDR
VQDC	1.50	104%
MTI	1.55	101%

Table 4.5: Performance of KD-GAD compared with kd-tree K-Means

Datasets	SpeedUp	SDR
VQDC	2.79	110%
MTI	8.40	102%

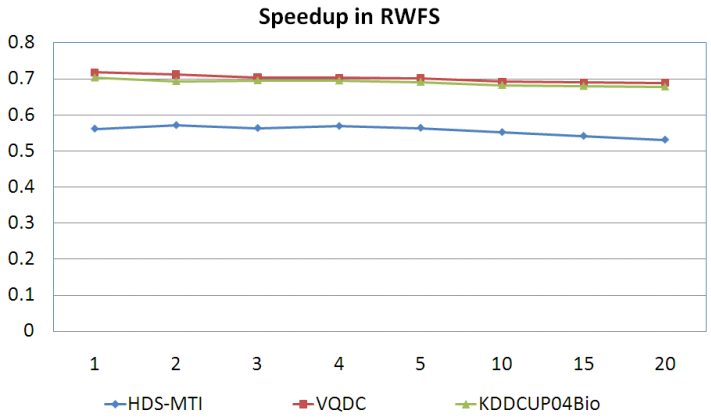


Figure 4.24: Speed degradation by RWFS, measured with Speedup. The horizontal axis denotes the value of R, and the vertical axis denotes the Speedup value.

impacts the performance of RWFS, we do experiments on it.

Figures 4.24 and 4.25 show the results on Speedup and SDR on several datasets. Because RWFS performs a global full search at the Rth iteration, it will degrade the speed. However, with the help of RWFS, WB-AGAD can lead to better cluster quality. Setting R as about 10 can achieve good performance. For dataset VQDC, the best improvement in SDR is 5%, HDS-MTI is 3% and KDDCUP04Bio is 4%. In situations that the clustering quality is more important than speed, we can use RWFS.

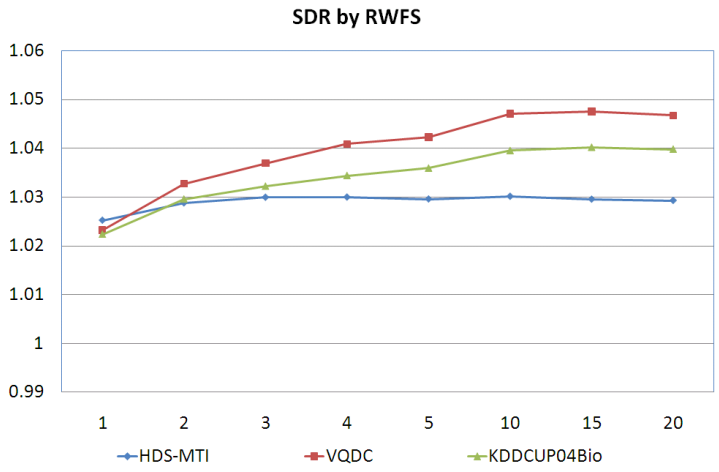


Figure 4.25: Clustering quality improvement by RWFS, measured with SDR. The horizontal axis denotes the value of R, and the vertical axis denotes the SDR value.

4.7 Discussion and Conclusion

4.7.1 Generality of the GAD Framework

The GAD framework is *general* due to the following properties:

- It works for both exact and approximate fast clustering.
- It is the general solution to exploit activity detection for fast clustering. GAD handles any m nearest centers. One advantage is that with the increase of m, GAD is able to make Full Search Area as small as possible. Another advantage is that it makes GAD capable of performing fast and high quality approximate clustering.
- It is flexible to embrace any distance measures, both metric and non-metric. Many fast clustering strategies, such as TIE [19], MPS [86] and Elkan [9], only work for metric distances. Non-metric distances are also very useful in many applications [44].
- Many other fast clustering strategies can be integrated with GAD to further improve their speed. [58] shows activity detection can speed up PDS [8], TIE [19] and

MPS [86]. [63] demonstrates activity detection can speed up MFAUPI, which is an extension of FAUPI [62]. Integrating all possible existing methods with GAD is out of the scope of this study; however, we have provided some examples, such as kd-tree [81] and hierarchical clustering [76].

4.7.2 Cluster Centers Initialization

Many initializing methods [93] can be used by GAD to get initial cluster centers, such as choosing the first K data patterns, random partitions, density based initialization [4], Intelligent initialization [72], iterative application of principal components [95], reverse nearest neighbor (RNN) search [117] and furthest first initialization.

The density based initialization [4] approach works as follows. Define D_a as the average pairwise distance among the set of patterns. For each pattern i, the density is defined as the number of patterns that are within D_a of i. The pattern with the highest density is picked as the first initial center, and the remaining K centers are chosen by decreasing density, as long as they are not within a pre-defined distance threshold (e.g. D_a) to any existing initial centers.

The approach introduced in [95] iteratively applies principal components to find K initial centers hierarchically. Beginning with the whole set of data pattern x_i as a single cluster, map to the first principle direction y_i. Partition the cluster into two sub-clusters C_1 and C_2 by putting patterns with $y_i \leq \bar{y}$ in C_1 and others in C_2. Pick the sub-cluster which has the largest within-cluster variance as the second cluster to be partitioned. Repeat until K clusters are found.

The reverse nearest neighbor (RNN) search [117] approach sorts the data patterns by decreasing order of the number of reverse nearest neighbors, defined as the set of patterns whose nearest neighbor is the target pattern, and picks the first pattern and its RNNs as an initial candidate and cluster. Repeat on remaining patterns until the list is empty. The

final K centers are chosen from the candidates by a RFN criterion.

Initialization Methods and GAD. A good initialization can make the iterative clustering converge faster; however, since GAD can save computation related to static centers and some active centers at each iteration, GAD can always achieve speedup. Different initializations may result in different initial activity percentage, thus have some impact on the Speedup. (Note that the activity percentage is also impacted by other factors such as the number of clusters, data dimensions, data type and the inherit data clustering structure.) In this study, we do initialization using the widely used random partitions method in our experiments. It is a future work to evaluate the impact of every initialization method (there are over 12 methods described in [93]).

As shown in experiments of datasets with sizes vary from 30,000 to over one million, GAD is faster than K-means. Even for using subsample, which can be used as either an initialization method or an approximate method, we can also run GAD instead of K-means on such subsample to still get speedup over K-means both on the subsample itself and on the whole dataset in general.

4.7.3 Conclusion

We propose a General Activity Detection (GAD) framework for fast clustering. We show that GAD is the general solution for activity detection based fast clustering. Two existing algorithms GT and CGAUCD are special cases of the GAD framework.

Within the GAD framework, we propose exact algorithm E-GAD. It is several times faster than K-Means and the best Speedup can be as high as 10 times. We can safely use E-GAD instead of K-Means and GT, because E-GAD is faster, gets the same result, has almost the same space complexity, and is easy to integrate other techniques. E-GAD is also faster than CGAUCD.

With different assumptions and levels of approximation, we propose four approximate

GAD algorithms: NS-AGAD, S-AGAD, I-AGAG and WB-AGAD. All of them are faster than E-GAD. I-AGAD is easy to achieve Speedup and very high clustering quality. WB-AGAD is the fastest and can achieve Speedup over E-GAD as high as 25 times within 98% clustering quality.

For clustering with very large clusters, we demonstrate that within the GAD framework, H-GAD and KD-GAD are better than their counterpart algorithms HKM and kd-tree K-Means, both in speed and clustering quality.

Method RWFS is introduced to improve the quality of approximate clustering. We can also integrate many other existing clustering quality improving methods, such as Swapping and Clustering Ensemble, with GAD.

The most important aspect of GAD is that it provides the general solution to exploit activity detection for fast clustering and our proposed algorithms within the framework can achieve very high speed. Many other fast clustering strategies can be further speeded up by GAD.

Future Works. There could be many future studies regarding how to extend GAD. We discuss several extensions to the GAD framework, including *Parallel GAD* which performs parallel GAD based clustering for extremely large datasets, *Automatic K GAD* which automatically selects the best number of clusters, and *Stream GAD* which performs fast GAD clustering on data stream. More specifically,

1. **Parallel GAD**. For extremely large datasets which do not fit in the memory, we can extend GAD to parallel computing on distributed systems. Inherent to GAD framework is intrinsic parallelism. The most intensive calculation is the distances between patterns and centers. Evenly partition the dataset among processes while the cluster centers are replicated. Save the nearest neighbors for each pattern at each process and perform fast GAD clustering in parallel. Communicate the updated centers and avoid the static centers to save the cost.

2. **Automatic decision of K for GAD**. Perform automatic clusters number decision within the GAD. Finding the best clusters number K^* involves comparing the clustering results of different number of clusters. The framework proposed in [82] specifies a range in which K^* reasonably lies and search for K^* which scores best by a criterion such as Bayesian Information Criterion (BIC). To make this more efficient in the GAD, instead of performing independent clusterings for each clusters number, we utilize the nearest neighbors from last clustering and only update those are impacted in splitting (or merging), thus speedup the clustering procedures.

3. **Compute GAD in Stream**. Extend GAD to fast clustering of data stream. Keep the current activity states of the cluster centers, update the cluster centers when new instances come. The nature of data stream makes the cluster centers frequently updated, many centers will easily become active and thus limit the advantage of using activity detection. So a suitable strategy is to adopt ε-Approximate Static Center instead of Static Center to keep many centers approximately static.

4.8 Systematic Applications

This section describes two systematic applications of the GAD algorithm in our social media demo systems, LikeMiner [49] and SocialSpamGuard [48].

4.8.1 LikeMiner

'**Like**' has recently become a very popular *social* function on the Internet. Many social media websites, such as Facebook, Twitter, YouTube and Flickr, provide the like/favorite button to allow users to express their like of the *objects* (such as text messages, comments, webpages, photos and videos) posted either by personal users or companies and public figures. Many traditional websites now also include the 'like' button, either as a plugin

from social media websites or created by themselves, to help promote the webpages. Figure 4.26 shows the like/favorite buttons used by Facebook, YouTube, theAtlantic, Amazon and Flickr.

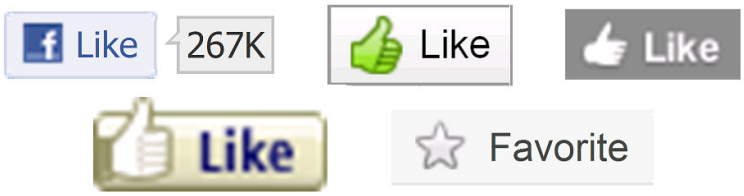


Figure 4.26: The like/favorite button for Facebook, YouTube, theAtlantic, Amazon and Flickr, respectively.

If a user clicks 'like' associated with an object, this directly indicates that s/he is highly interested in the object. So the 'like' function provides a more accurate way of estimating user interests than non-direct indicators, such as user-service interaction [121]. Additionally, in some social networks, e.g., Facebook, when a user clicks 'like' to an object, such action will be immediately shared to his/her friends (under allowed privacy setting). So the 'like' function also provides a useful and effective way of sharing or promoting information in social media. Actually, sharing by 'like' may have higher influence because people may pay more attention to the objects *liked* than simply *shared* by friends.

We proposed a system called LikeMiner to mine the power of 'like' in social media networks. As shown in Figure 4.28, the system architecture works as follows. (1) Forming a network model for social media with likes. Figure 4.27 shows an example of such network within Facebook. A user can post an object to his/her wall, or to any company/public-figure pages s/he is a fan. Such object can be liked by his/her friends or non-friends who are also fans of that page. A page can also post its own objects and those interesting ones will be liked by its fans. (2) Extract the topic distribution for the objects, both visual and textual. (3) Perform link mining based on the network structure and the object topic distribution. (4) Web interface for browsing, keyword-based topic query and interest-based recommendation, such as recommending product photos or company

pages to users who may potentially like them.

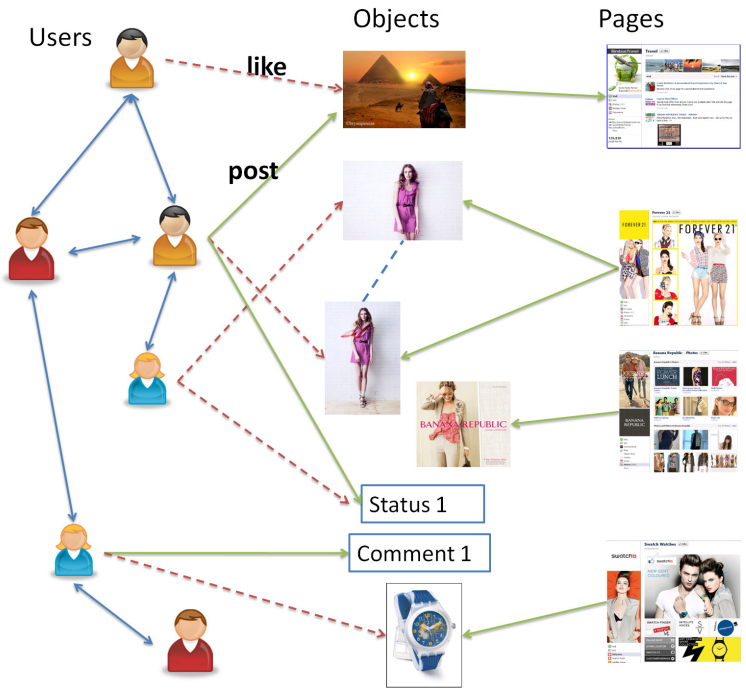


Figure 4.27: A heterogeneous network model for social media with 'like' function, using Facebook as an example. A blue bidirectional arrow is the friendship link. A red dashed arrow is a *like* action, while a green arrow denotes a *post* action, annotated with the time stamp when it was posted. The dashed blue line shows that the two photos are visually similar.

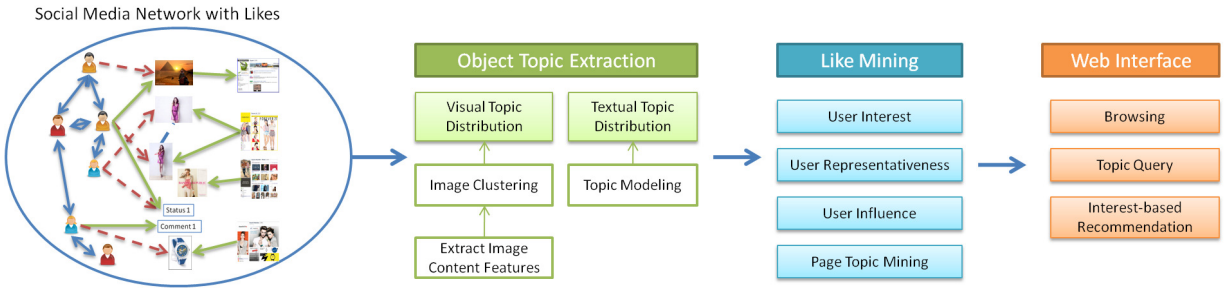


Figure 4.28: LikeMiner System Architecture.

GAD was used in the Visual Topic Extraction component of the system. More specifically, we used GAD to perform large scale clustering of the photos for each feature space, and treat each cluster as a visual topic. The topic distribution of a photo is based on the similarity of the photo to each cluster center.

4.8.2 SocialSpamGuard

Social media websites allow users to freely distribute and share information to friends. Information can spread very fast and easily within the social media networks. Because of this, such websites expose to various types of unwanted and malicious spammer or hacker actions. There is a crucial need in the society and industry [1] for a security solution in social media. Social media websites need to be clean for long term success. A company/brand page on social media also needs to be clean to reduce the risk of damaging its reputation. Virus links from the spams could lead to personal or business loss and damage.

There have been some studies on detecting spam emails [61, 125], spam messages [91], spam images [16], spam video [11], web spam [112], spammers [71] [64] [111], etc. However, one of the major challenges of spam detection in social media is that the spams are usually in the form of photos and text, and in the context of large scale dynamic social network. We need a comprehensive solution which can consider text, photos and the social network features, and also be scalable and capable of performing real-time detection.

We proposed *SocialSpamGuard*, a scalable and online social media spam detection system based on data mining for social network security. The major advantages of the proposed approach can be summarized as follows:

1. Automatically harvesting spam activities in social network by monitoring social sensors with popular user bases;

2. Introducing both image and text content features and social network features to indicate spam activities;

3. Integrating with our GAD clustering algorithm to handle large scale data;

4. Introducing a scalable active learning approach to identify existing spams with

[1] Defensio. http://defensio.com

limited human efforts, and perform online active learning to detect spams in real-time.

As shown in Figure 4.29, we model a typical social media network as a *time-stamped heterogeneous information network* $G = \langle V, E \rangle$. V is the set of different types of nodes, such as users (U), pages (P) and posts (Q) (including text description and/or images/videos (I), with the time stamp). E denotes the set of links between nodes, for example, friendship/following links between users, fan/favorite links between users and pages. Images are indirectly-linked together by content similarity (dashed lines).

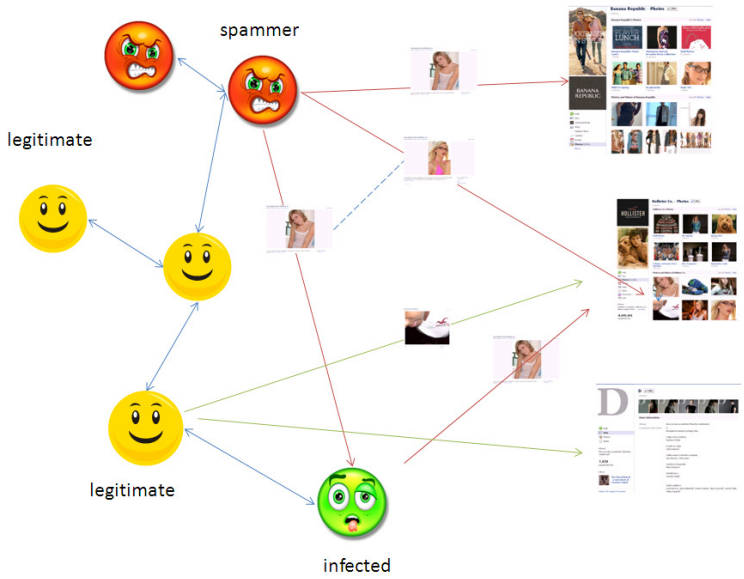


Figure 4.29: Heterogeneous Information Network for Social Media. A red face is a spammer, a yellow smile face is a legitimate user, a yellow face turned to green color is an infected user. The blue directed line is the friendship/following link. A red arrow is a spam post, while a green arrow is a ham post.

Posting is one of the predominant user activities in social media. People spend a lot of time in social media, such as Facebook/Twitter, on posting or checking the posts of friends or favorite pages. The posts can be generally labeled as two categories: **spam** (unwanted, irrelevant, promotional or harmful social posts) and **ham** (legitimate social posts). There are three types of users: spammer, legitimate user and infected user. Infected user are legitimate user who send spams after being infected by virus. Our **goal** is to identify the

spam posts sent from spammers and infected users.

As shown in Figure 4.30, the system architecture works as follows. In the first stage, we collect historical social media data, extract both content (including text and images) and social network features, perform active learning to build classification model and identify spams. In the second stage, we monitor the real-time activity of the social network and perform online active learning, make prediction and send alarms to clients about detected spams, collect feedbacks from clients and update the model.

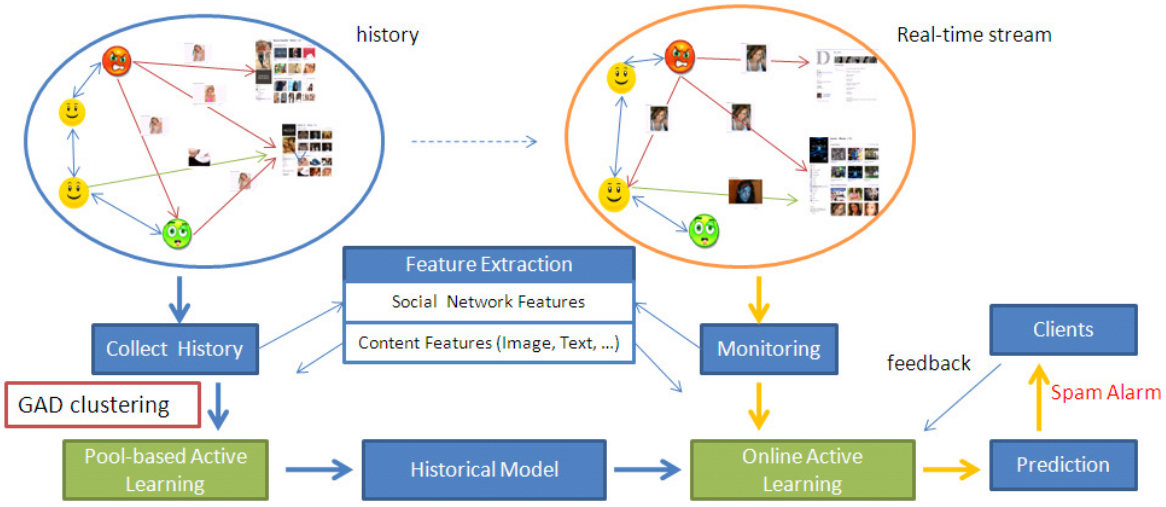


Figure 4.30: SocialSpamGuard System Architecture.

Active learning was the core algorithm to detect spams in this system. Because of the huge number of posts, randomly sampling may not be a good choice due to the uneven distribution and duplicate (or near duplicate) posts. To generate a smarter sample in the active learning procedure, we used GAD to perform large scale clustering of the posts into large number of clusters and make sampling from the clusters to increase diversity and avoid duplicates.

Chapter 5

Social Media for Prediction: A Unified Regression Model that Integrates Topic-based Clustering and User Ranking

Social media content not only relate to each other, but also to outside phenomena and show strong implication for prediction. In this section, by aggregating user content information, we propose a unified model to integrate clustering, ranking and regression for prediction. This work extends the content relation from within the domain to external domain, by considering the relation between social media content objects and outside variables.

5.1 Introduction

Both governments and industries are interested in social trends. For example, politicians use polling to measure their popularity for elections and to monitor public sentiment to decide which position to take on social issues. Industry polls potential consumers to understand product acceptance. Although it is an expensive undertaking to perform polling, it is an investment that is critical for organizations both large and small to use for resource allocation and planning.

Social media makes the "The Wisdom of Crowds" [98] much easier to obtain, because it provides a good platform to explore the global trends and sentiments that can be drawn by analyzing the sharing patterns of uploaded and downloaded social media. For example, each time a content object, such as a comment, image or video, is published or viewed, it constitutes an implicit vote for (or against) the subject of the content. This vote carries

along with it a rich set of associated data including time and (often) location information. By aggregating such votes across millions of Internet users, we can capture the wisdom that is embedded in social media sites for applications such as politics, economics, finance and marketing.

Many work shows that the actions of individual Internet users, when properly pooled, can indicate macro trends. There are studies using Search Engine queries for influenza Internet surveillance [21], such as Google Trends [36], search advertisement click through [31], Yahoo search queries [84] and health website access logs [52].

Study on user web access logs from the Healthlink Web site [51] showed that there is a moderately strong correlation between the number of influenza-related article accesses and the CDC surveillance data.

Gunther [31] showed that there is a correlation between the number of clicks on keyword ”flu” or ”flu symptoms” triggered sponsored link in Google AdSense (appeared for Canadian searchers only) with epidemiological data from the flu season 2004/2005 in Canada.

Specifically in [36], Google search engine queries and data from the Centers for Disease Control (CDC) are used to find 45 specific search terms that are related to the percentage of influenze related physician visits. This model allows for monitoring influenza rates 1-2 weeks *ahead* of the CDC reports.

The problem of using general search engines is that the original query log is not publicly available and the query trends may become noisy under the impact of news events. For example, as soon as a new product is announced by a major technology company, blogs will begin to report and speculate about the product.

Joshua and Ewan [77] used prediction markets and Twitter to predict a swine flu pandemic. They ”explore the hypothesis that social media such as Twitter encodes the belief of a large number of people about some concrete statement about the world”. Such beliefs are aggregated using a Prediction Market specifically concerning the possibility of

a Swine Flu Pandemic in 2009. They show that features extracted from Tweets can reduce the error associated with modeling these beliefs. The approach outperforms baseline methods based purely on time-series information from the Market.

Aron Culotta [22] studied how to detect influenza outbreaks by analyzing Twitter messages. Over 500 million Twitter messages were analyzed from an eight month period. The result showed that by tracking a small number of flu-related keywords, we are able to forecast future influenza rates with high accuracy, with a 95% correlation with national health statistics.

In this study, we propose a unified model to integrate clustering, ranking and regression for prediction, with application for twitter based stock change prediction [88] [14].

Figure 5.1: An example of tweet about stock FB (Facebook Inc) posted by the user Sarah Frier on June 6.

For each *tweet* that is stock related, it is posted by some *user*, links to one or several *stocks*, and contains a bag of *words*, describing the users' opinions or information related to the listed stocks. A tweet mentioning stock symbol FB (Facebook Inc) is shown in Fig. 5.1. Each tweet is talking some specific aspects about the stocks. By viewing each tweet as a text document, we can use *topics*, i.e., a distribution over terms, to describe these different aspects.

For the stock price prediction task, we are particularly interested in predicting the daily stock price change percentage, which is denoted as y. An example of FB stock price change from May 24th to June 7th is shown in Fig. 5.2, where the daily percentage change is given in the bottom.

Figure 5.2: A 10-day change of FB stock price.

5.2 Problem Definition

In this study, our goal is to utilize the public buzz about a given stock in twitter, to predict the daily stock price change in terms of percentage. In order to predict the stock price change of the next day, we first collect all the tweets about the given stock in the previous day, and use features extracted from these tweets to predict the stock price by feeding them to the trained model.

The basic assumptions of how tweets can influence the stock price include:

1. **Clustering tweets into topics.** Some topics have a positive impact of the stock price, such as launching new product, while others have a negative impact of the stock price, such as revue sliding. Therefore, these topics can be used as a good feature set to predict the stock price.

2. **Rank the tweets by user credibility.** The credibility of the tweets diverges a lot, and depends on the users' credibility. Some of the users get more accurate information or have more influences than others. Therefore, each tweet should be assigned with different weights that are carrying the credibility score of their users.

3. **Regression based on the overall topic distribution.** The overall topic distribution of the tweets shows a summarized opinion of the users, which provide a valuable information for stock prediction.

More importantly, the topics of the tweets and the credibility of the users in turn should be guided and automatically learned by the prediction model. Without guidance, there are numerous different ways to generate reasonable topics given a corpus of documents. However, topics generated unsupervisedly may not be discriminative enough for a particular stock prediction task. Therefore, we need to use supervised topic model to generate the topic space that is helpful. Similarly, the credibility of the user should be learned by whether the user can always produce trustable and influential information that leads to good predicting accuracy for a specific stock.

In all, our goal is to predict the stock price change, by building a supervised predicting model that unifies (1) topic-based clustering for tweets, (2) credibility ranking for tweets/users, and (3) stock price change prediction into a single probabilistic model.

A probabilistic model is proposed in Section 5.3, and a learning algorithm is proposed in Section 5.4. From the algorithm, we can see that the three aspects of the model, namely clustering, ranking and prediction, are mutually enhance each other. The topics achieved by clustering is enhanced by correctly assigning the credibility score to each tweet and user and a better regression model; the ranking of tweets and users in terms of credibility is enhanced when a better clustering and regression result are achieved; and the regression accuracy is further enhanced when the clustering and ranking are both good.

5.3 Unified Generative Model

In this section, we propose a unified generative model for our stock price prediction task. The graphical model is shown in Figure 5.3, and the notations are summarized in Table 5.1.

For a given stock and given date (called a case), the stock price change rate is denoted as y, which is an observed response variable in the range of real number R. We assume y is sampled from a linear regression model, with residuals following Gaussian distribution.

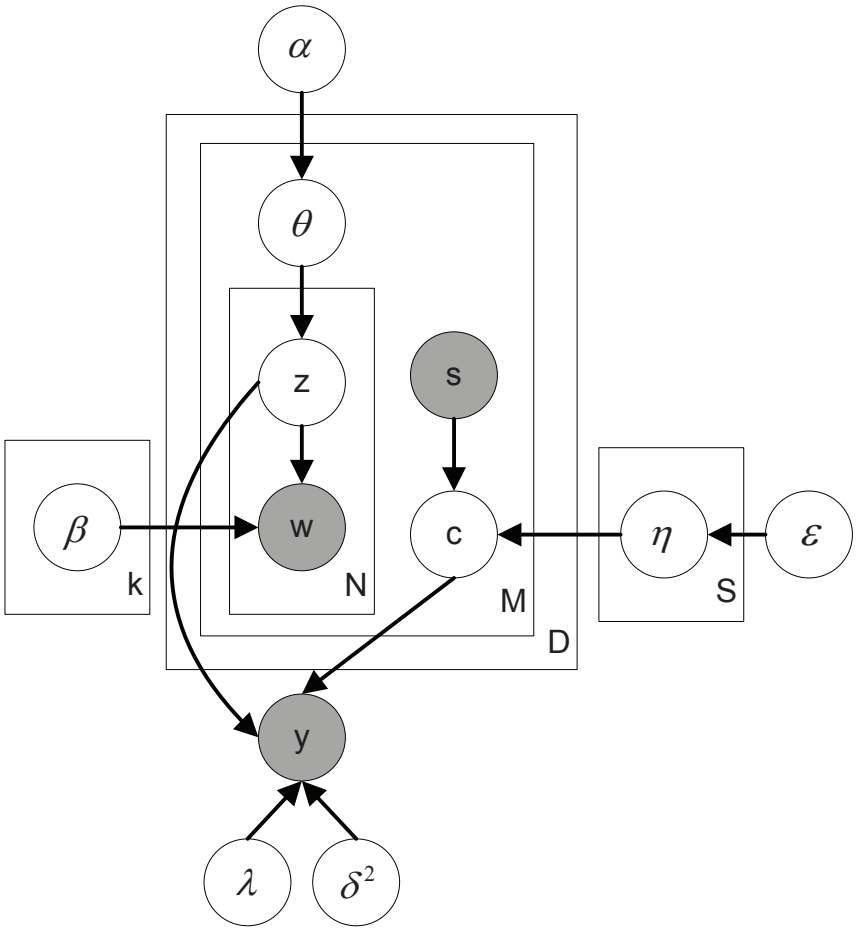


Figure 5.3: Graphical Model Representation of the Proposed Model.

Table 5.1: Notation Summarization

Notation	Meaning
i	tweet/document index
N_i	number of unique terms in document i
j	unique term index
M	number of tweets for the current stock
M_d	number of tweets in case d, and d is a date-company pair
V	number of unique terms in vocabulary
K	number of clusters/topics
z_{ij}	the topic label of term j of tweet i
c	binary credibility for the tweet
S	number of users/sources
s	integer 0, 1, ..., S, user/source id
η_s	probability parameter for Bernoulli distribution for user s
s_i	the user id for document i
α	a K-dimensional parameter vector for Dirichlet prior
β	$K \times V$ matrix, β_k is word distribution for topic k
λ	K-dimensional coefficient vector associate with K topics
δ^2	variance
ϵ	a 2-dimensional parameter vector for Beta prior

We collect all the tweets related to the given stock within a given time range. Suppose there are K topics described as $\beta_{1:K}$, and we have the Dirichlet parameter vector α for the topic distribution, the parameters for the linear regression model λ and δ^2, the user for each tweet, and the gamma prior parameter vector ϵ for Bernoulli distribution, then the credibility scores for users, the tweets, and the response variables are generated in the following way:

1. For each user s
   (a) Draw credibility sore $\eta_s|\epsilon \sim Beta(\epsilon)$.
2. For each tweet i
   (a) Draw topic distribution $\theta_i|\alpha \sim Dir(\alpha)$.
   (b) For each term
      i. Draw topic label z_{ij} for term j by $z_{ij}|\theta_i \sim Mult(\theta_i)$.

ii. Draw term w_j by $w_j|z_{ij} \sim Mult(\beta_{z_{ij}})$.

(c) For the observed user s

i. Draw binary credibility score c_i for the tweet by $c|\eta_s \sim Bernoulli(\eta_s)$

3. Draw stock price change rate $y|z_{ij}, c_i, \lambda, \delta^2 \sim N(\lambda^T \frac{1}{M} \sum_{i=1}^{M} c_i \bar{z}_i, \delta^2)$, where $\bar{z}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} z_{ij}$.

The joint probability defined by Figure 5.3 can be formalized as:

$$
\begin{aligned}
& p(\mathbf{w}, \mathbf{y}, \mathbf{z}, \theta, \mathbf{c}, \mathbf{s}, \eta | \alpha, \beta, \lambda, \delta^2, \epsilon) \\
&= \prod_{s}^{S} p(\eta_s|\epsilon) p(y|\lambda^{\mathsf{T}} \frac{1}{M} \sum_{i}^{M} c_i \bar{\mathbf{z}}_\mathbf{i}, \delta^2) \prod_{i}^{M} p(c_i|\eta_{s_i}) p(\theta_i|\alpha) \prod_{j}^{N_i} p(w_{ij}|\beta_{z_{ij}}) p(z_{ij}|\theta_i) \qquad (5.1)
\end{aligned}
$$

The goal is then to estimate all the parameters, $\alpha, \beta, \lambda, \delta^2$, and ϵ to maximize the joint probability.

5.4 Learning Algorithm

5.4.1 Bounded Approximation

By Jenson's inequality we can derive the **lower bound** of the data log-likelihood as,

$$
L(\mathbf{w}, y, \mathbf{s}) \geq E_q[\log p(\mathbf{w}, y, \mathbf{z}, \theta, \mathbf{c}, \mathbf{s}, \eta | \alpha, \beta, \lambda, \delta^2, \epsilon)] - E_q[\log q(\mathbf{z}, \theta, \mathbf{c}, \eta | \gamma, \phi, \pi, \sigma)]
$$

If we consider the whole corpus of D cases, then the overall lower bound is

$$
L(\mathbf{w}, \mathbf{y}, \mathbf{s}) \geq \sum_{d}^{D} \left\{ E_q[\log p(\mathbf{w}, y, \mathbf{z}, \theta, \mathbf{c}, \mathbf{s}, \eta | \alpha, \beta, \lambda, \delta^2, \epsilon)] - E_q[\log q(\mathbf{z}, \theta, \mathbf{c}, \eta | \gamma, \phi, \pi, \sigma)] \right\} \quad (5.2)
$$

where $q(\mathbf{z}, \theta, \mathbf{c}, \eta | \gamma, \phi, \pi, \sigma)$ is the proposal distribution for the variational inference.

In particular, we use the mean-field approximation to relax the dependency among the latent variables, and define the proposal distribution as,

$$q(\mathbf{z}, \theta, \mathbf{c}, \eta|\gamma, \phi, \pi, \sigma) = \prod_{s}^{S} q(\eta_s|\sigma_s) \prod_{i}^{M} q(c_i|\pi_i) q(\theta_i|\gamma_i) \prod_{j}^{N_i} q(z_{ij}|\phi_{ij}) \tag{5.3}$$

where $q(\eta|\sigma)$ is Beta distribution $Beta(\sigma_0, \sigma_1)$ (σ is a 2-dimensional vector for each source), $q(c|\pi)$ is Bernoulli distribution, i.e., $q(c = 1) = \pi$ (π is a real number for each document between 0 and 1), $q(\theta|\gamma)$ is Dirichlet distribution $Dir(\gamma)$ (γ is a k-dimensional for each document), and $q(z|\phi)$ is Multinomial distribution $Mul(\phi)$ (ϕ is a k-dimensional vector for each word in each document).

By expanding Eq (5.2) we get the following lower bound approximation:

$$\begin{aligned}
& L(\mathbf{w}, y, \mathbf{s}) \\
\geq & E_q[\log p(y|\lambda^\top \frac{1}{M} \sum_{i}^{M} c_i \bar{\mathbf{z}}_\mathbf{i}, \delta^2)] + \sum_{s}^{S} E_q[\log p(\eta_s|\epsilon)] \\
& + \sum_{i}^{M} E_q[\log p(\theta_i|\alpha)] + \sum_{i}^{M} E_q[\log p(c_i|\eta_{s(i)})] \\
& + \sum_{i}^{M} \sum_{j}^{N_i} \sum_{l}^{k} E_q[\log p(w_{ij}|\beta_{z_{ij}=l})] + \sum_{i}^{M} \sum_{j}^{N_i} \sum_{l}^{k} E_q[\log p(z_{ij} = l|\theta_i)] \\
& - \sum_{s}^{S} E_q[\log q(\eta_s|\sigma_s)] - \sum_{i}^{M} E_q[\log q(c_i|\pi_i)] \\
& - \sum_{i}^{M} E_q[\log q(\theta_i|\gamma_i)] - \sum_{i}^{M} \sum_{j}^{N_i} \sum_{l}^{k} E_q[\log q(z_{ij} = l|\phi_{ij})]
\end{aligned} \tag{5.4}$$

The object of EM algorithm is to maximize the righthand side of Eq (5.4) with respect to the variational parameters $\Theta_v = \{\gamma, \phi, \pi, \sigma\}$ (E-Step) and model parameters $\Theta_m = \{\alpha, \beta, \lambda, \delta^2, \epsilon\}$ (M-Step).

The expectation terms can be calculated as follows,

$$
\begin{aligned}
& E_q[\log p(y|\lambda^\top \frac{1}{M}\sum_i^M c_i \bar{\mathbf{z}}_\mathbf{i}, \delta^2)] \\
= & -\frac{1}{2}\log\delta^2 - \frac{y^2}{2\delta^2} + \frac{y}{\delta^2}\lambda^\top(\frac{1}{M}\sum_i^M \frac{\pi_i}{N_i}\sum_j^{N_i}\phi_{ij}) \\
& - \frac{1}{2\delta^2 M^2}\sum_i^M \frac{\pi_i}{N_i^2}\lambda^\top\Big(\sum_{j_1}^{N_i}\sum_{j_2\neq j_1}^{N_i}\phi_{ij_1}\phi_{ij_2}^\top + \sum_j^{N_i} Diag\{\phi_{ij}\}\Big)\lambda \\
& - \frac{1}{2\delta^2 M^2}\sum_{i_1=1}^M\sum_{i_2\neq i_1}^M \frac{\pi_{i_1}\pi_{i_2}}{N_{i_1}N_{i_2}}\lambda^\top\Big(\sum_{j_1=1}^{N_{i_1}}\sum_{j_2=1}^{N_{i_2}}\phi_{i_1 j_1}\phi_{i_2 j_2}^\top\Big)\lambda
\end{aligned} \tag{5.5}
$$

$$
E_q[\log p(\theta_i|\alpha)] = \log\Gamma(\sum_l \alpha_l) - \sum_l \log\Gamma(\alpha_l) + \sum_l(\alpha_l - 1)[\Psi(\gamma_{il}) - \Psi(\sum_j \gamma_{ij})] \tag{5.6}
$$

where $\Gamma(\cdot)$ is the gamma function and $\Psi(\cdot)$ is the first order derivative of log gamma function, which is equal to digamma/gamma.

$$
E_q[\log q(\theta_i|\gamma_i)] = \log\Gamma(\sum_l \gamma_{il}) - \sum_l \log\Gamma(\gamma_{il}) + \sum_l(\gamma_{il} - 1)[\Psi(\gamma_{il}) - \Psi(\sum_j \gamma_{ij})] \tag{5.7}
$$

$$
E_q[\log p(w_{ij}|\beta_{z_{ij}=l})] = \phi_{ijl} w_{ij} \log \beta_{z_{ij}=l, v_{ij}} \tag{5.8}
$$

where v_{ij} means the jth word in ith document is word v_{ij} in the vocabulary, z_{ij} means the corresponding topic assignment is z_{ij}.

$$
E_q[\log p(z_{ij} = l|\theta_i)] = \phi_{ijl}[\Psi(\gamma_{il}) - \Psi(\sum_e \gamma_{ie})] \tag{5.9}
$$

$$E_q[\log q(z_{ij} = l|\phi_{ij})] = \phi_{ijl} \log \phi_{ijl} \tag{5.10}$$

$$E_q[\log p(c_i = 1|\eta_{s(i)})] = \pi_i[\Psi(\sigma_{s(i)1}) - \Psi(\sigma_{s(i)0} + \sigma_{s(i)1})] \tag{5.11}$$

$$E_q[\log p(c_i = 0|\eta_{s(i)})] = (1 - \pi_i)[\Psi(\sigma_{s(i)0}) - \Psi(\sigma_{s(i)1} + \sigma_{s(i)0})] \tag{5.12}$$

$$E_q[\log q(c_i|\pi_i)] = \pi_i \log \pi_i + (1 - \pi_i) \log(1 - \pi_i) \tag{5.13}$$

$$E_q[\log p(\eta_s|\epsilon)] = \log \Gamma(\epsilon_0 + \epsilon_1) - \sum_j \log \Gamma(\epsilon_j) + \sum_j (\epsilon_j - 1)[\Psi(\sigma_{sj}) - \Psi(\sum_e \sigma_{se})] \tag{5.14}$$

$$E_q[\log q(\eta_s|\sigma_s)] = \log \Gamma(\sigma_{s0} + \sigma_{s1}) - \sum_j \log \Gamma(\sigma_{sj}) + \sum_j (\sigma_{sj} - 1)[\Psi(\sigma_{sj}) - \Psi(\sum_e \sigma_{se})] \tag{5.15}$$

where $s(i)$ means the source index for ith document.

With the above expectation terms, EM algorithm would just reduce to Maximization-Maximization alternations with respect to Θ_v and Θ_m.

5.4.2 Parameters Estimation

E-step Parameters

This section shows how to estimate E-step parameters: $\Theta_v = \{\gamma, \phi, \pi, \sigma\}$

Parameter γ

Similar to LDA, we can derive the formula for updating parameter γ as:

$$\gamma_{ij} = \alpha_j + \sum_{e=1}^{N_i} \phi_{iej} \tag{5.16}$$

If we consider $d \in D$, the formula becomes:

$$\gamma_{dij} = \alpha_j + \sum_{e=1}^{N_{di}} \phi_{diej} \tag{5.17}$$

Parameter ϕ

Denote the object function related to ϕ_{ij} as $L(\phi_{ij})$, and we can find it is only related to Eq (5.8), (5.9), (5.10) and (5.5). Therefore, the first order derivative of $L(\phi_{ij})$ with respect to ϕ_{ijl} can be calculated as,

$$\begin{aligned}
&\frac{\partial L(\phi_{ij})}{\partial \phi_{ijl}} \\
=&w_{ij} \log \beta_{z_{ij=l}, v_{ij}} + [\Psi(\gamma_{il}) - \Psi(\sum_e \gamma_{ie})] - \log \phi_{ijl} - 1 \\
&+ \frac{y\pi_i}{\delta^2 N_i M} \lambda_l - \frac{\pi_i}{2\delta^2 N_i^2 M^2} \Big[2\lambda^\intercal \sum_{k \neq j}^{N_i} \phi_{ik} \lambda + \lambda \circ \lambda \Big]_l \\
&- \frac{2}{2\delta^2 M^2} \sum_{i_2 \neq i}^{M} \frac{\pi_i \pi_{i_2}}{N_i N_{i_2}} \Big(\lambda^\intercal \sum_{j_2=1}^{N_{i_2}} \phi_{i_2 j_2} \lambda_l \Big)
\end{aligned} \tag{5.18}$$

with respect to the constrain that $\phi_{ijl} \geq 0$ and $\sum_{l=1}^{k} \phi_{ijl} = 1$. $\circ$ means the position multiply for vectors.

As a result, the updating equation for ϕ_{ijl} is,

$$\begin{aligned}
&\log \phi_{ijl} \\
\propto & w_{ij} \log \beta_{z_{ij}=l, v_{ij}} + \Psi(\gamma_{il}) + \frac{y\pi_i}{\delta^2 N_i M}\lambda_l \\
& - \frac{\pi_i}{2\delta^2 N_i^2 M^2}\Big(2(\lambda^\top \sum_{k\neq j}^{N_i} \phi_{ik})\lambda_l + \lambda_l^2\Big) \\
& - \frac{2}{2\delta^2 M^2}\sum_{i_2\neq i}^{M} \frac{\pi_i \pi_{i_2}}{N_i N_{i_2}}\Big(\lambda^\top \sum_{j_2=1}^{N_{i_2}} \phi_{i_2 j_2}\lambda_l\Big)
\end{aligned} \tag{5.19}$$

where $[\cdot]_l$ means the lth element in this vector.

Parameter π

The first order derivative of $L(\pi)$ with respect to π is,

$$\begin{aligned}
&\frac{\partial L(\pi_i)}{\partial \pi_i} \\
=&[\Psi(\sigma_{s(i)1}) - \Psi(\sigma_{s(i)0} + \sigma_{s(i)1})] - [\Psi(\sigma_{s(i)0}) - \Psi(\sigma_{s(i)1} + \sigma_{s(i)0})] \\
& - \{log\pi_i + 1 - log(1-\pi_i) - 1\} \\
& + \frac{y}{\delta^2}\lambda^\top \frac{1}{M}\frac{1}{N_i}\sum_{j=1}^{N_i}\phi_{ij} \\
& - \frac{1}{2\delta^2 M^2}\frac{1}{N_i^2}\lambda^\top\Big(\sum_{j_1}^{N_i}\sum_{j_2\neq j_1}^{N_i}\phi_{ij_1}\phi_{ij_2}^\top + \sum_{j}^{N_i} Diag\{\phi_{ij}\}\Big)\lambda \\
& - \frac{2}{2\delta^2 M^2}\sum_{i_2\neq i}^{M}\frac{\pi_{i_2}}{N_i N_{i_2}}\lambda^\top\Big(\sum_{j_1=1}^{N_i}\sum_{j_2=1}^{N_{i_2}}\phi_{ij_1}\phi_{i_2 j_2}^\top\Big)\lambda
\end{aligned} \tag{5.20}$$

Set $\frac{\partial L(\pi_i)}{\partial \pi_i} = 0$, we get

$$
\begin{aligned}
& log\frac{\pi_i}{1-\pi_i} \\
=& \Psi(\sigma_{s(i)1}) - \Psi(\sigma_{s(i)0}) \\
& + \frac{y}{\delta^2 M N_i}\lambda^\intercal \sum_{j=1}^{N_i}\phi_{ij} \\
& - \frac{1}{2\delta^2 M^2}\frac{1}{N_i^2}\lambda^\intercal\Big(\sum_{j_1}^{N_i}\sum_{j_2\neq j_1}^{N_i}\phi_{ij_1}\phi_{ij_2}^\intercal + \sum_{j}^{N_i} Diag\{\phi_{ij}\}\Big)\lambda \\
& - \frac{1}{\delta^2 M^2}\sum_{i_2\neq i}^{M}\frac{\pi_{i_2}}{N_i N_{i_2}}\lambda^\intercal\Big(\sum_{j_1=1}^{N_i}\sum_{j_2=1}^{N_{i_2}}\phi_{ij_1}\phi_{i_2 j_2}^\intercal\Big)\lambda \\
& = G
\end{aligned} \tag{5.21}
$$

The update formula for π_i is

$$
\pi_i = \frac{e^G}{1+e^G} \tag{5.22}
$$

Parameter σ

Following the description of A.3 in LDA paper, we can get the estimate for σ as,

$$
\sigma_{s0} = \sum_{d}^{D}\sum_{i}^{M_d}(1-\pi_{di})\Delta[s(di)=s] + \epsilon_0 \tag{5.23}
$$

$$
\sigma_{s1} = \sum_{d}^{D}\sum_{i}^{M_d}\pi_{di}\Delta[s(di)=s] + \epsilon_1 \tag{5.24}
$$

M-step Parameters

This section shows how to estimate M-step parameters: $\Theta_m = \{\alpha, \beta, \lambda, \delta^2, \epsilon\}$.

Parameter α

We first derive the follow two formulas:

$$\frac{\partial L(\alpha_i)}{\partial \alpha_i} = \sum_{j=1}^{M} \left\{ \Psi(\sum_{l}^{K} \alpha_l) - \Psi(\alpha_i) + \Psi(\gamma_{ji}) - \Psi(\sum_{l=1}^{K} \gamma_{jl}) \right\}$$
$$= M[\Psi(\sum_{l}^{K} \alpha_l) - \Psi(\alpha_i)] + \sum_{j=1}^{M} [\Psi(\gamma_{ji}) - \Psi(\sum_{l=1}^{K} \gamma_{jl})] \tag{5.25}$$

$$\frac{\partial L(\alpha)}{\partial \alpha_l \partial \alpha_m} = M\Psi'(\sum_{e} \alpha_e) - \delta(l,m) M\Psi'(\alpha_l) \tag{5.26}$$

If we consider $d \in D$, they become:

$$\frac{\partial L(\alpha_j)}{\partial \alpha_j} = \sum_{d=1}^{D} \sum_{i=1}^{M_d} \left\{ \Psi(\sum_{l}^{K} \alpha_l) - \Psi(\alpha_j) + \Psi(\gamma_{dij}) - \Psi(\sum_{l=1}^{K} \gamma_{dil}) \right\}$$
$$= \sum_{d=1}^{D} M_d [\Psi(\sum_{l}^{K} \alpha_l) - \Psi(\alpha_j)] + \sum_{d=1}^{D} \sum_{i=1}^{M_d} [\Psi(\gamma_{dij}) - \Psi(\sum_{l=1}^{K} \gamma_{dil})] \tag{5.27}$$

$$\frac{\partial L(\alpha)}{\partial \alpha_l \partial \alpha_m} = \sum_{d=1}^{D} M_d \left\{ \Psi'(\sum_{e} \alpha_e) - \delta(l,m) \Psi'(\alpha_l) \right\} \tag{5.28}$$

Use the linear-time Newton-Raphson algorithm described in LDA to compute the values.

Parameter β

Similar to LDA, we can derive the formula as:

$$\beta_{lv} \propto \sum_{i} \sum_{j} \phi_{ijl} w_{ij}^{v} \tag{5.29}$$

If we consider $d \in D$, it becomes:

$$\beta_{lv} \propto \sum_{d}^{D} \sum_{i}^{M_d} \sum_{j}^{N_{d_i}} \phi_{dijl} w_{dij}^{v} \tag{5.30}$$

where w_{ij}^{v} is the count for word v.

Based on the above relation, β can be computed as

$$\beta_{lv} = \frac{\sum_i \sum_j \phi_{ijl} w_{ij}^{v}}{\sum_i \sum_j \sum_v \phi_{ijl} w_{ij}^{v}} \tag{5.31}$$

If consider $d \in D$, it becomes:

$$\beta_{lv} = \frac{\sum_d \sum_i \sum_j \phi_{dijl} w_{dij}^{v}}{\sum_d \sum_i \sum_j \sum_v \phi_{dijl} w_{dij}^{v}} \tag{5.32}$$

Parameter λ

Denote the object function related to λ as $L(\lambda)$, which only associates with Eq (5.5),

$$\begin{aligned}
& L(\lambda) \\
= & \sum_d \{ \frac{y_d}{\delta^2} \lambda^{\mathsf{T}} (\frac{1}{M_d} \sum_i^{M_d} \frac{\pi_{di}}{N_{di}} \sum_j^{N_{di}} \phi_{dij}) \\
& - \sum_i^{M_d} \frac{\pi_{di}}{2\delta^2 N_{di}^2 M_d^2} \lambda^{\mathsf{T}} \Big(\sum_j^{N_{di}} \sum_{k \neq j}^{N_{di}} \phi_{dij} \phi_{dik}^{\mathsf{T}} + \sum_j^{N_{di}} Diag\{\phi_{dij}\} \Big) \lambda \\
& - \frac{1}{2\delta^2 M_d^2} \sum_{i_1=1}^{M_d} \sum_{i_2 \neq i_1}^{M_d} \frac{\pi_{di_1} \pi_{di_2}}{N_{di_1} N_{di_2}} \lambda^{\mathsf{T}} \Big(\sum_{j_1=1}^{N_{di_1}} \sum_{j_2=1}^{N_{di_2}} \phi_{di_1 dj_1} \phi_{di_2 dj_2}^{\mathsf{T}} \Big) \lambda \}
\end{aligned} \tag{5.33}$$

And the first order derivative of $L(\lambda)$ with respect to λ is,

$$\begin{aligned}
&\frac{\partial L(\lambda)}{\partial \lambda} \\
&= \sum_d \{ \frac{y_d}{\delta^2 M_d} (\sum_i^{M_d} \frac{\pi_{di}}{N_{di}} \sum_j^{N_{di}} \phi_{dij}) - \sum_i^{M_d} \frac{\pi_{di}}{\delta^2 N_{di}^2 M_d^2} \Big(\sum_j^{N_{di}} \sum_{k \neq j}^{N_{di}} \phi_{dij} \phi_{dik}^\intercal + \sum_j^{N_{di}} Diag\{\phi_{dij}\} \Big) \lambda \} \\
&- \sum_d \frac{2}{2\delta^2 M_d^2} \sum_{i_i=1}^{M_d} \sum_{i_2 \neq i_1}^{M_d} \frac{\pi_{di_1} \pi_{di_2}}{N_{di_1} N_{di_2}} \Big(\sum_{j_1=1}^{N_{di_1}} \sum_{j_2=1}^{N_{di_2}} \phi_{di_1 dj_1} \phi_{di_2 dj_2}^\intercal \Big) \lambda
\end{aligned} \tag{5.34}$$

By setting Eq (5.34) to be zero, we get,

$$\begin{aligned}
&\tilde{\lambda} \\
&= \Big[\sum_d \frac{y_d}{\delta^2 M_d} (\sum_i^{M_d} \frac{\pi_{di}}{N_{di}} \sum_j^{N_{di}} \phi_{dij}) \Big] \\
&\Big[\sum_d \sum_i^{M_d} \frac{\pi_{di}}{\delta^2 N_{di}^2 M_d^2} \Big(\sum_j^{N_{di}} \sum_{k \neq j}^{N_{di}} \phi_{dij} \phi_{dik}^\intercal + \sum_j^{N_{di}} Diag\{\phi_{dij}\} \Big) \\
&+ \sum_d \frac{2}{2\delta^2 M_d^2} \sum_{i_i=1}^{M_d} \sum_{i_2 \neq i_1}^{M_d} \frac{\pi_{di_1} \pi_{di_2}}{N_{di_1} N_{di_2}} \Big(\sum_{j_1=1}^{N_{di_1}} \sum_{j_2=1}^{N_{di_2}} \phi_{di_1 dj_1} \phi_{di_2 dj_2}^\intercal \Big) \Big]^{-1}
\end{aligned} \tag{5.35}$$

Parameter δ^2

The first order derivative of $L(\delta^2)$ with respect to δ^2 is,

$$\begin{aligned}
&\frac{\partial L}{\partial \delta^2} \\
&= -\frac{1}{2\delta^2} + \frac{y^2}{2(\delta^2)^2} - \frac{y}{(\delta^2)^2} \lambda^\intercal (\frac{1}{M} \sum_i^M \frac{\pi_i}{N_i} \sum_j^{N_i} \phi_{ij}) \\
&+ \frac{1}{2(\delta^2)^2 M^2} \sum_i^M \frac{\pi_i}{N_i^2} \lambda^\intercal \Big(\sum_{j_1}^{N_i} \sum_{j_2 \neq j_1}^{N_i} \phi_{ij_1} \phi_{ij_2}^\intercal + \sum_j^{N_i} Diag\{\phi_{ij}\} \Big) \lambda \\
&+ \frac{1}{2(\delta^2)^2 M^2} \sum_{i_1=1}^M \sum_{i_2 \neq i_1}^M \frac{\pi_{i_1} \pi_{i_2}}{N_{i_1} N_{i_2}} \lambda^\intercal \Big(\sum_{j_1=1}^{N_{i_1}} \sum_{j_2=1}^{N_{i_2}} \phi_{i_1 j_1} \phi_{i_2 j_2}^\intercal \Big) \lambda
\end{aligned} \tag{5.36}$$

Let $\frac{\partial L}{\partial \delta^2} = 0$, we get the update formula as

$$
\begin{aligned}
&\delta^2 \\
=&y^2 - 2y\lambda^\mathsf{T}(\frac{1}{M}\sum_i^M \frac{\pi_i}{N_i}\sum_j^{N_i}\phi_{ij}) \\
&+ \frac{1}{M^2}\sum_i^M \frac{\pi_i}{N_i^2}\lambda^\mathsf{T}\Big(\sum_{j_1}^{N_i}\sum_{j_2\neq j_1}^{N_i}\phi_{ij_1}\phi_{ij_2}^\mathsf{T} + \sum_j^{N_i} Diag\{\phi_{ij}\}\Big)\lambda \\
&+ \frac{1}{M^2}\sum_{i_1=1}^M\sum_{i_2\neq i_1}^M \frac{\pi_{i_1}\pi_{i_2}}{N_{i_1}N_{i_2}}\lambda^\mathsf{T}\Big(\sum_{j_1=1}^{N_{i_1}}\sum_{j_2=1}^{N_{i_2}}\phi_{i_1j_1}\phi_{i_2j_2}^\mathsf{T}\Big)\lambda
\end{aligned} \tag{5.37}
$$

Parameter ϵ

Related to LDA, the estimation for ϵ can be based on,

$$\frac{\partial L(\epsilon)}{\partial \epsilon_l} = S[\Psi(\sum_e \epsilon_e) - \Psi(\epsilon_l)] + \sum_s^S[\Psi(\sigma_{sl}) - \Psi(\sum_e \sigma_{se})] \tag{5.38}$$

$$\frac{\partial L(\epsilon)}{\partial \epsilon_l \partial \epsilon_m} = S\Psi'(\sum_e \epsilon_e) - \delta(l,m)S\Psi'(\epsilon_l) \tag{5.39}$$

And then we use the Newton-Raphson algorithm to update the values.

5.5 Prediction

In prediction phase, since we don't know the y label beforehand, we can only depend on the observed document content w to infer the topic distribution and then make the prediction.

More specifically, we would modify Eq (5.10) to be,

$$\log \phi_{ijl}^{\text{new}} \propto \left\{w_{ij}\log \beta_{z_{ij}=l,v_{ij}} + \Psi(\gamma_{il}^{\text{new}})\right\} \tag{5.40}$$

The inference procedure for γ_i^{new} is the same as we did in the training phase. Then the

prediction can be calculated as,

$$E[Y|\mathbf{w}, \alpha, \beta, \lambda, \delta^2, \epsilon] \approx \lambda^\intercal \frac{1}{M} \sum_i^M \frac{\pi_i}{N_i} \sum_j^{N_i} \phi_{ij}^{\text{new}} \tag{5.41}$$

where π_i can be computed as

$$\pi_i = \begin{cases} \frac{\sigma_{s(i)1}}{\sigma_{s(i)0}+\sigma_{s(i)1}} & \text{if } s(i) \text{ exists in training set} \\ \frac{\epsilon_1}{\epsilon_0+\epsilon_1} & \text{otherwise} \end{cases} \tag{5.42}$$

5.6 Experiment

In this section, we present experiment results of our model and compare with baselines.

5.6.1 Dataset and Preprocessing

Our dataset contains two parts: one is the Twitter social media data and the other is stock price data. The twitter data was downloaded from Twitter API (http://www.twitter.com) from February 27, 2012 to June 1, 2012 (96 days). The daily stock price data during the same period for all the companies (around 6000) in NYSE, NASDAQ and AMEX was downloaded from Yahoo Finance (http://finance.yahoo.com).

We obtained real-time twitter data for those stock companies. There are over 2 million tweets in our dataset that mention the stock company symbols (defined by the corresponding stock exchange market), such as $AAPL for Apple and $GOOG for Google. We use the stock price change percentage of each company at each day as the response variable for prediction and the set of tweets about the company (which means the tweets contain the company symbol) to learn the model.

We filtered out tweets that are retweeting of other people's tweet and focus on original opinions. In order to achieve this goal, we use some text analysis rules, such as ignore tweets that begin with "RT".

5.6.2 Performance Result

This section presents prediction result of our model, and meanwhile, shows the topic-based clusters and user ranking result.

Prediction Performance

We evaluate the prediction performance of the compared algorithms by two measures: **Direction-Accuracy** and **RMSE**. The last 10 days in the dataset are used as testing set and previous days as training set.

Direction-Accuracy tells the percentage of predictions that match correctly with the stock change direction (either increase or decrease). In order to compute this score, we map the output of our model to either increase or decrease based on the sign of the prediction.

Since Direction-Accuracy essentially measures performance on classification, we compare our model with popular classification algorithms, including SVM, Naive Bayes, RIPPER (a rule-based learning algorithm) and Decision Tree. For SVM, we tried three major kernels: linear, polynomial and radial basis function (RBF). Their performance are similar, but RBF kernel is the best. So we only report result for RBF-SVM. Following traditional approaches that use news to predict stock, we represent the tweets as space vector on the terms, each dimension is the term frequency over the set of tweets for each case. Then use the space vector to feed to the four baseline algorithms for training.

Figure 5.4 shows the accuracy of the algorithms. Transitional wisdom about stock market prediction is that stock price change is too random to be predictable and it is very hard for any model to achieve better than 50% accuracy. Our result on the traditional approaches show that with the help of social media data, the algorithms can achieve better than 50% prediction accuracy, but still lower than 60%. CRR is the proposed model in this work, and CRRnoR is a simplified version that ignore user ranking. We can see

that user ranking play an important role in the market direction prediction task. Our proposed model can make prediction that is much better than baseline algorithms with over 66% direction-accuracy. The reason is that our model can capture topic clusters and user credit ranking in an integrated way that can help extract more useful information for prediction.

Traditional approaches work on the terms dimension; however, users can express similar idea with different terms. For example, if a user feels a stock will increase tomorrow, different terms, such as ”buy”, ”long”, ”bullish” can be used to express such opinion. If we can perform topic-based clustering of the tweets, we will be able to know the percentage of users that talk about this topic, and learn prediction model based on the topic level, instead of the term level. Social media data is big but so noisy, some users may have higher creditability than others. If we can perform user ranking to give more weight to higher ranking users in our model, better performance can be achieved.

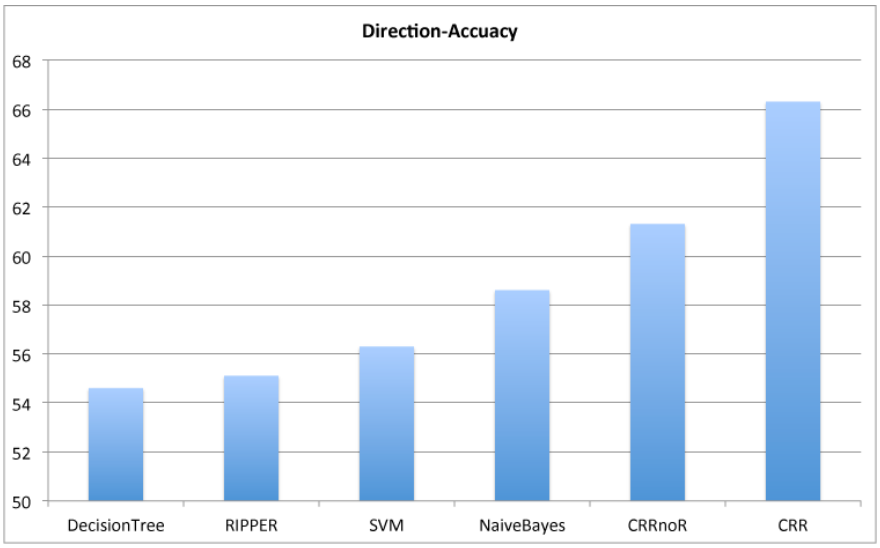


Figure 5.4: Accuracy of the algorithms

RMSE (Root-Mean-Square Error) measures how closely a regression model can predict the value of stock change percentage. Since exact price change percentage is too volatile to predict, we partition it into 5 ordered levels: -2:($-\infty$, -0.05] , -1:(-0.05, 0), 0, -1:(0, 0.05),

2:[0.05, ∞). More than 5% change in stock market is generally treated as a big change, so we believe this ordered level is meaningful. We compare our model to two popular regression algorithms: SVM regression (SMOreg) and linear regression (LinerReg). Figure 5.5 shows the RMSE of the algorithms. The result shows that our model achieves lower error rate than existing algorithms.

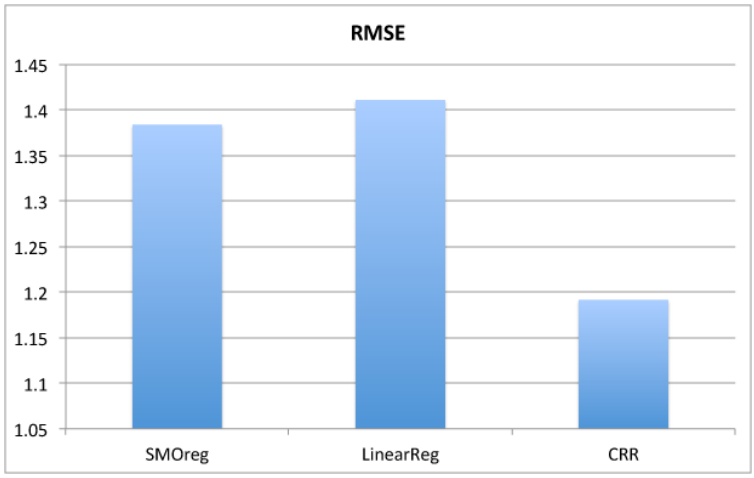


Figure 5.5: RMSE of the algorithms

Topic-based Clusters

Table 5.2 shows top 2 clusters that has the highest absolute λ value and their top ranking topic terms. Cluster 1 has the highest negative λ value, which means this topic is negatively correlated with the target variable, i.e., the stock change. Cluster 2 has the highest positive λ value, most terms in this group are positive about stocks. We can see that our model can automatically discover such discriminative groups that can be used for prediction.

User Ranking

Table 5.3 shows the top ranked twitter users. We can see that the ranking result makes a lot of sense, because most top ranked users are professional stock traders.

Table 5.2: Top topic clusters with top ranked topic terms.

Cluster 1	Cluster 2
down	bullish
bearish	buy
sold	up
sell	long
drops	bull
short	higher
bad	buying
lower	holding
loss	growth
bottom	love
bear	profit
fall	hold
fuck	breakout
falling	bot
downgraded	puts
worst	high
damn	bounce

5.7 Conclusion and Future Work

Social media data can become very valuable when we are able to extract implication from them for real case problem. Our work of using social media for stock market prediction is an great example. In this work, we proposed a unified generative framework to integrate topic-based clustering, user ranking and regression for the task of prediction based on user content information. The advantage of our model is that it can simultaneously output cluster of terms that form topic groups that are discriminative for prediction, ranking of user credibility and achieve better prediction performance.

We have conducted experiment on three months of twitter data and the corresponding daily stock data of all companies listed on the three major stock exchange markets (NYSE, NASDAQ and AMEX) in USA. Experiment result shows that our model can achieve over 66% direction-accuracy which is much better than traditional learning methods, and lower regression error rate compared with popular regression algorithms. Our result show that

Table 5.3: Top ranked users. The description is extracted from the public user profile information on Twitter in June 2012. Note that the copyright of each description belongs to the corresponding Twitter user.

Twitter Username	Description
InvestorEntryPt	Tackling the stock market one great trade at a time. Technical Swing Trader
joey_volpe	Wharton/U of Pennsylvania Alum, Technology/Stocks/Space Enthusiast (Business Opinion Account).
HCPG	Active traders with focus on commodities and tech
apppro1	LONG-term INVESTOR looking for ways to build our Nation
downtowntrader	Active Trader blogging on my site, downtowntrader.com. frequent contributor to Investopedia.com and chartadvisor.com.
PrimoPennies	Penny Stock and Options Alerts. www.primopennies.us
RockyBIP	Biotech trader. Founder, The Biotech Investment Paradigm.
SchtickyTrader	I like to schtick to the long side and swing my schtick when the feeling's right.
tradefast	Corporate Chief Investment Officer with 26+ yrs of professional money management experience with major hedge funds & financial institutions
sharkfoot	Franchise Player. I trade.
_thewiseman	Day trader
iequityresearch	Founded by Western Expats working and living around the World. Impartial & Independent ADR research is what we are all about.
KlickStocks	Posting & following specific, measurable stock picks
mstrades	engineer, entrepreneur & stock trader. Main strategies are to short overbought stocks and buy technical breakouts/bounces. Target: $10k to $1 mill in 10 years

while traditional wisdom usually think stock is not predictable, however, with the help of our technology to analyze the implication of user generated content information in social media, we are in better position than before when trying to make decision no the risky stock market.

Future Work The proposed model can be extended in different ways, for example: (1) Extend the current model to consider external features, such as, company features (market cap, EPS, number of shares, price per share, P/E ratio, etc.) and historical prices & volumes. This can be easily achieved by adding additional features to the regression part

of the formula to extend the dimension of parameter λ. (2) Extend the model to consider company network. Companies in the same industry section, or competitors, are likely to show correlations (could be either positive or negative) between their stock prices. By considering such learned and domain knowledge of the company relationships, we can introduce a regulation factor based on such relation to enhance our model for better performance.

Chapter 6

Summary

Social media websites allow users to create and upload user-generated content and provides convenient ways to let them interact and collaborate with each other, in contrast to Web 1.0 websites where users are limited to the passive viewing of content that was created for them. Examples of social media websites include social networks (Facebook, LinkedIn, MySpace), blogs (Blogger) and microblogs (Twitter), wikis (Wikipedia), forums, reviews and opinions (Digg), social tagging (or social bookmarking) (Delicious), multimedia sharing (Flickr and YouTube), Social News, prediction markets, virtual worlds, online chatting (AIM, MSN, GTalk, QQ) and social online games. The phenomenal success of social media sites, such as Facebook, Twitter, LinkedIn, Flickr and YouTube, not only revolutionized the way people communicate and think, but also revolutionized the way how corporations do business.

In this thesis, I advance the data mining technique to mine the large scale user generated content in social media to explore the hidden relations and implications. Such analysis can be very useful for business, government, as well as individuals.

One basic problem for content relation mining is how to compute content similarity in the social media setting. Content similarity computation is very important for many reasons, for example, grouping together similar content can help better aggregate trends of topics for prediction, identifying similar content objects can help recommendation in social media. We propose an efficient approach called MoK-SimRank to significantly improve the speed of SimRank, which utilizes the network structure, and introduce its extension HMok-SimRank to work on weighted heterogeneous information networks in

social media. Then we propose algorithm IWSL to provide a novel way of integrating both link and content information. IWSL performs content and link reinforcement style learning with either global or local feature weight learning.

Given any similarity measure, data clustering can partition similar content objects into groups and provide a compact representation of the content relations. One big challenge for clustering such content objects in social media is the large scale data. To deal with the problem, we propose a GAD (General Activity Detection) framework to fully explore the power of activity detection for clustering. We design a set of algorithms within this framework for fast clustering in different scenarios. The most important contribution of our work is that GAD is the general solution to exploit activity detection for fast clustering and our algorithms within the framework can achieve very high speed.

Relation between the IWSL and GAD. IWSL can be used to learning better feature weighting with the help of link information. Then we can use the weighted feature vector for GAD clustering, which could potentially achieve better clustering quality.

Social media content not only relate to each other, but also to outside phenomena and show strong implication with prediction power. For example, both governments and industries are interested in social trends, and users' tweets about the companies also imply their opinions and could be used for stock price change prediction. By aggregating user content information, we developed a unified model to integrate clustering, ranking and regression for prediction.

Relation between network based content similarity and the soical media based prediction framework. In order to improve the prediction framework, as in the stock prediction case, one extension is to consider the relation between companies. Such relation can be automatically obtained by using the network based content similarity computation method.

Relation between GAD and the prediction framework. While we have used topic-based clustering in our framework, due to the popularity of topic modeling based approach for text analysis, it is also possible to extend GAD to be supervised by outside variable for

guided clustering, then GAD could also be used in our framework.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[2] Selim Aksoy and Robert M. Haralick. Textural features for image database retrieval. In *CBAIVL '98: Proceedings of the IEEE Workshop on Content - Based Access of Image and Video Libraries*, page 45, Washington, DC, USA, 1998. IEEE Computer Society.

[3] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. OPTICS: ordering points to identify the clustering structure. *SIGMOD Record*, 28(2):49–60, 1999.

[4] M.M. Astrahan. Speech analysis by clustering, or the hyperphome method. *Stanford Artificial Intelligence Project Memorandum AIM-124, Stanford, CA: Stanford University*, 1970.

[5] Hanan G. Ayad and Mohamed S. Kamel. Cumulative voting consensus method for partitions with variable number of clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(1):160–173, 2008.

[6] Boris Babenko, Steve Branson, and Serge Belongie. Similarity functions for categorization: from monolithic to category specific. In *International Conference on Computer Vision (ICCV'09)*, Kyoto, Japan, 2009.

[7] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[8] Chang-Da Bei and R. M. Gray. An improvement of the minimum distortion encoding algorithm for vector quantization. *IEEE Transactions on Communication*, 33:1132–1133, October 1985.

[9] Florian Beil, Martin Ester, and Xiaowei Xu. Using the triangle inequality to accelerate k-means. In *Twentieth International Conference on Machine Learning (ICML'03)*, pages 147–153, 2003.

[10] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[11] Fabrício Benevenuto, Tiago Rodrigues, Virgilio Almeida, Jussara M. Almeida, Chao Zhang, and Keith W. Ross. Identifying video spammers in online social networks. In *AIRWeb*, pages 45–52, 2008.

[12] Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, April 1980.

[13] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 97–104, New York, NY, USA, 2006. ACM.

[14] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

[15] Marinette Bouet and Chabane Djeraba. Powerful image organization in visual retrieval systems. In *MULTIMEDIA '98: Proceedings of the sixth ACM international conference on Multimedia*, pages 315–322, New York, NY, USA, 1998. ACM.

[16] Byungki Byun, Chin-Hui Lee, Steve Webb, and Calton Pu. A discriminative classifier learning approach to image modeling and spam image identification. In *CEAS*, 2007.

[17] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, and Sridhar Rajagopalan. Automatic resource compilation by analyzing hyperlink structure and associated text. In *7th World Wide Web Conference (WWW'97)*, pages 65–74, 1997.

[18] Savvas Chatzichristofis and Yiannis Boutalis. CEDD: Color and edge directivity descriptor: A compact descriptor for image indexing and retrieval. *Lecture Notes in Computer Science*, pages 312–322, 2008.

[19] Sin-Horng Chen and W. M. Hsieh. Fast algorithm for vq codebook design. In *Proceedings Inst. Elect. Eng.*, volume 138, pages 357–362, October 1991.

[20] Rudi L. Cilibrasi and Paul M. B. Vitanyi. The Google similarity distance. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):370–383, 2007.

[21] Courtney D. Corley and Armin R Mikler. A computational framework to study public health epidemiology. In *International Joint Conferences on System Biology, Bioinformatics and Intelligent Computing (IJCBS09)*, Shanghai, China, August 2009.

[22] Aron Culotta. Detecting influenza outbreaks by analyzing twitter messages, 2010. cite arxiv:1007.4748.

[23] M. Datar, N. Immorlica, P. Indyk, and V.Mirrokni. Locality sensitive hashing scheme based on p-stable distribution. In *Proceeding of Twentitieth Annual Symposium on Computational Geometry*, 2004.

[24] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2):1–60, April 2008.

[25] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[26] Thomas Deselaers, Daniel Keysers, and Hermann Ney. Features for image retrieval: an experimental comparison. *Information Retrieval*, 11(2):77–107, 2008.

[27] Thomas Deselaers and Henning Mller. Combining textual- and content-based image retrieval, tutorial. In *The 19th International Conference on Pattern Recognition*, ICPR'08, 2008.

[28] Comaniciu Dorin and Meer Peter. Mean Shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[29] Sandrine Dudoit and Jane Fridlyand. Bagging to improve the accuracy of a clustering procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[30] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[31] Gunther Eysenbach. Infodemiology: tracking flu-related searches on the web for syndromic surveillance. In *AMIA 2006 Symposium Proceedings*, pages 244–248, 2006.

[32] Gary W. Flake, Robert E. Tarjan, and Kostas Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[33] Daniel Fogaras and Balazs Racz. Scaling link-based similarity search. In *Proceedings of the 14th International Conference on World Wide Web*, pages 641–650, New York, NY, USA, 2005. ACM Press.

[34] D. Frossyniotis, A. Likas, and A. Stafylopatis. A clustering method based on boosting. *Pattern Recognition Letters*, 25:641–654, 2004.

[35] T. Gevers and A.W.M. Smeulders. PicToSeek: Combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1):102–119, January 2000.

[36] Jeremy Ginsberg, Matthew H. Mohebbi, Rajan S. Patel, Lynnette Brammer, Mark S. Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014, 2009.

[37] William I. Grosky. Multimedia information systems. *IEEE MultiMedia*, 1(1):12–24, 1994.

[38] Venkat N. Gudivada and Vijay V. Raghavan. Content-based image retrieval systems. *Computer*, 28:18–22, 1995.

[39] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, Washington, DC, USA, 1999. IEEE Computer Society.

[40] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, second edition, March 2006.

[41] Alexander Hinneburg and Daniel A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 58–65, 1998.

[42] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 762, Washington, DC, USA, 1997. IEEE Computer Society.

[43] Tektronix Inc. Normalized contour as a shape descriptor for visual objects. *ISO/IEC/JTC1/SC29/WG11, Lancaster, UK, Feb*, page 579, 1999.

[44] David W. Jacobs, Daphna Weinshall, and Yoram Gdalyahu. Classification with nonmetric distances: image retrieval and class representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):583–600, June 2000.

[45] G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining(KDD'02)*, 2002.

[46] Yu-Gang Jiang, Chong-Wah Ngo, and Jun Yang. Towards optimal bag-of-features for object categorization and semantic video retrieval. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, pages 494–501, New York, NY, USA, 2007. ACM.

[47] Xin Jin, Sangkyum Kim, Jiawei Han, Liangliang Kao, and Zhijun Yin. GAD: General activity detection for fast clustering on large data. In *2009 SIAM Int. Conf. on Data Mining (SDM'09)*, pages 2–13, Sparks, NV, USA, April 2009.

[48] Xin Jin, Cindy Xide Lin, Jiebo Luo, and Jiawei Han. Socialspamguard: A data mining-based spam detection system for social media networks. *PVLDB*, 4(12):1458–1461, 2011.

[49] Xin Jin, Chi Wang, Jiebo Luo, Xiao Yu, and Jiawei Han. Likeminer: a system for mining the power of 'like' in social media networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 753–756, New York, NY, USA, 2011. ACM.

[50] Yushi Jing and Shumeet Baluja. VisualRank: Applying pagerank to large-scale image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1877–1890, 2008.

[51] Heather A. Johnson, Michael M. Wagner, William R. Hogan, Wendy Chapman, Robert T Olszewski, John Dowling, and Gary Barnas. Analysis of web access logs for surveillance of influenza. *Studies in Health Technology and Informatics*, 107(Pt 2):1202–C1206, 2004.

[52] Heather A. Johnsona, Michael M. Wagnera, William R. Hogana, Wendy Chapmana, Robert T Olszewskia, John Dowlinga, and Gary Barnas. Analysis of web access logs for surveillance of influenza. *Stud Health Technol Inform.*, 107(Pt 2):1202C6, 2004.

[53] Ian T Jolliffe. *Principal component analysis*. Springer-Verlag, 2002.

[54] Dhiraj Joshi, James Z. Wang, and Jia Li. The story picturing engine—a system for automatic text illustration. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):68–89, 2006.

[55] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry: Theory and Applications*, 28:89–112, 2004.

[56] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *Computer*, 32(8):68–75, 1999.

[57] Leonard Kaufman and Peter J. Rousueeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[58] Timo Kaukoranta, Pasi Fränti, and Olli Nevalainen. A fast exact gla based code vector activity detection. *IEEE Transactions on Image Processing*, 9(8):1337–1342, 2000.

[59] KDDCUP04. Kddcup 04 biology dataset. http://kodiak.cs.cornell.edu/kddcup/datasets.html (accessed 2008), 2008.

[60] W. Y. Kim and Y. S. Kim. A rotation invariant geometric shape descriptor using zernike moment. *ISO/IEC/JTC1/SC29/WG11, Lancaster, UK, Feb*, page 687, 1999.

[61] Joseph S. Kong, Behnam Attaran Rezaei, Nima Sarshar, Vwani P. Roychowdhury, and P. Oscar Boykin. Collaborative spam filtering using e-mail networks. *IEEE Computer*, 39(8):67–73, 2006.

[62] Jim Z. C. Lai and Yi-Ching Liaw. Fast-searching algorithm for vector quantization using projection and triangular inequality. *IEEE Transactions on Image Processing*, 13(12):1554–1558, 2004.

[63] Jim Z. C. Lai, Yi-Ching Liaw, and Julie Liu. A fast vq codebook generation algorithm using codeword displacement. *Pattern Recognition*, 41(1):315–319, 2008.

[64] Kyumin Lee, James Caverlee, and Steve Webb. Uncovering social spammers: social honeypots + machine learning. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 435–442, 2010.

[65] Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, and Denis Turdakov. Accuracy estimate and optimization techniques for SimRank computation. *VLDB Endowment*, 1(1):422–433, 2008.

[66] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[67] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2*, page 1150, Washington, DC, USA, 1999. IEEE Computer Society.

[68] Ricoh Company Ltd. MINDS's descriptors for still images - spatial edge distribution descriptor. *ISO/IEC/JTC1/SC29/WG11, Lancaster, UK, Feb*, page 109, 1999.

[69] Wei-Ying Ma. *NETRA: a toolbox for navigating large image databases*. PhD thesis, Santa Barbara, CA, USA, 1998.

[70] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proceeding of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[71] Benjamin Markines, Ciro Cattuto, and Filippo Menczer. Social spam detection. In *AIRWeb*, pages 41–48, 2009.

[72] Boris Mirkin. *Clustering for Data Mining: A Data Recovery Approach*. London: Chapman and Hall, 2005.

[73] K. Muller and J. R. Ohm. Wavelet-based contour descriptor. *ISO/IEC/JTC1/SC29/WG11, Lancaster, UK, Feb.*, page 567, 1999.

[74] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[75] Carlton W. Niblack, Ron Barber, Will Equitz, Myron D. Flickner, Eduardo H. Glasman, Dragutin Petkovic, Peter Yanker, Christos Faloutsos, and Gabriel Taubin. QBIC project: querying images by content, using color, texture, and shape. volume 1908, pages 173–187. SPIE, 1993.

[76] David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, volume 2, pages 2161–2168, 2006.

[77] Joshua Rittermanand Miles Osborne and Ewan Klein. Using prediction markets and twitter to predict a swine flu pandemic. In *Proceedings of the 1st International Workshop on Mining Social Media*, 2009.

[78] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[79] Jeng-Shyang Pan, Zhe-Ming Lu, and Sheng-He Sun. An efficient encoding algorithm for vector quantization based on subvector technique. *IEEE Transactions on Image Processing*, 12(3):265–270, 2003.

[80] Mira Park, Jesse S. Jin, and Laurence S. Wilson. Fast content-based image retrieval using quasi-gabor filter and reduction of image feature dimension. In *SSIAI '02: Proceedings of the Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, page 178, Washington, DC, USA, 2002. IEEE Computer Society.

[81] D. Pelleg and A. Moore. Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of KDD'99*, pages 277–281, New York, NY, 1999. ACM.

[82] Dan Pelleg and Andrew W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[83] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 1–8, 2007.

[84] Philip M. Polgreen, Yiling Chen, David M. Pennock, and Forrest D. Nelson. Using internet searches for influenza surveillance. *Clinical Infectious Diseases (Supplement)*, pages 1443–1448, 2008.

[85] Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[86] S.-W. Ra and J.-K. Kim. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization. *IEEE Transactions on Circuits System*, 40:576–579, September 1993.

[87] Yong Rui, Thomas S. Huang, and Shih-Fu Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of Visual Communication and Image Representation*, 10(1):39–62, 1999.

[88] Eduardo J. Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 513–522, New York, NY, USA, 2012. ACM.

[89] Stan Sclaroff, Marco La Cascia, and Saratendu Sethi. Unifying textual and visual cues for content-based image retrieval on the world wide web. *Comput. Vis. Image Underst.*, 75:86–98, July 1999.

[90] Gholamhosein Sheikholeslami, Surojit Chatterjee, and Aidong Zhang. WaveCluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 8(3-4):289–304, 2000.

[91] Chandra Shekar, Shruti Wakade, Kathy J. Liszka, and Chien-Chung Chan. Mining pharmaceutical spam from twitter. In *ISDA*, pages 813–817, 2010.

[92] David McG. Squire, Wolfgang Muler, Henning Muler, and Thierry Pun. Content-based query of image databases: inspirations from text retrieval. *Pattern Recognition Letters*, 21(13-14):1193–1198, 2000. Selected Papers from The 11th Scandinavian Conference on Image.

[93] Douglas Steinley and Michael J. Brusco. Initializing k-means batch clustering: A critical evaluation of several techniques. *Journal of Classification*, 24(1):99–121, 2007.

[94] Alexander Strehl and Joydeep Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, 3:583–617, 2002.

[95] T. SU and J.G. DY. Another look at non-random methods for initializing k-means clustering. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, pages 784–786, 2004.

[96] Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. RankClus: integrating clustering with ranking for heterogeneous information network analysis. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 565–576, New York, NY, USA, 2009. ACM.

[97] Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 797–806, New York, NY, USA, 2009. ACM.

[98] J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.

[99] S.C. Tai, C.C. Lai, and Y.C. Lin. Two fast nearest neighbor searching algorithms for image vector quantization. *IEEE Transactions on Communication*, 44(12):1623–1628, 1996.

[100] Hideyuki Tamura and Naokazu Yokoya. Image database systems: A survey. *Pattern Recognition*, 17(1):29–43, 1984.

[101] Jinhui Tang, Haojie Li, Guo-Jun Qi, and Tat-Seng Chua. Image annotation by graph-based inference with integrated multiple/single instance representations. *IEEE Transactions on Multimedia*, 12(2):131–141, 2010.

[102] Antonio Torralba. 80 millioin tiny images. http://people.csail.mit.edu/torralba/tinyimages/. (accessed 2008), 2008.

[103] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: a large dataset for non-parametric object and scene recognition. Technical Report MIT-CSAIL-TR-2007-024, MIT, 2007.

[104] J. R. R. Uijlings, A. W. M. Smeulders, and R. J. H. Scha. Real-time bag of words, approximately. In *Proceeding of the ACM International Conference on Image and Video Retrieval*, CIVR '09, pages 6:1–6:8, New York, NY, USA, 2009. ACM.

[105] Luxburg Ulrike. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[106] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems: A survey. Technical report, Department of Computing Science, Utrecht University, 2002.

[107] Gang Wang and David Forsyth. Object image retrieval by exploiting online knowledge resources. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1–8, 2008.

[108] Jidong Wang, Hua-Jun Zeng, Zheng Chen, Hongjun Lu, Li Tao, and Wei-Ying Ma. ReCoM: reinforcement clustering of multi-type interrelated data objects. In *SIGIR*, pages 274–281. ACM, 2003.

[109] Shuo Wang, Feng Jing, Jibo He, Qixing Du, and Lei Zhang. IGroup: presenting web image search results in semantic clusters. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 587–596, New York, NY, USA, 2007. ACM.

[110] Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In *VLDB '97: Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[111] Steve Webb, James Caverle, and Calton Pu. Social honeypots: Making friends with a spammer near you. In *Proceeding of the Fifth Conference on Email and Anti-Spam (CEAS)*, 2008.

[112] Steve Webb, James Caverlee, and Calton Pu. Introducing the webb spam corpus: Using email spam to identify web spam automatically. In *Proceeding of the Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[113] Lei Wu, Steven C.H. Hoi, Rong Jin, Jianke Zhu, and Nenghai Yu. Distance metric learning from uncertain side information with application to automated photo tagging. In *Proceedings of the seventeen ACM International Conference on Multimedia*, pages 135–144, New York, NY, USA, 2009. ACM.

[114] Lei Wu, Xian-Sheng Hua, Nenghai Yu, Wei-Ying Ma, and Shipeng Li. Flickr distance. In *Proceeding of the 16th ACM International Conference on Multimedia*, pages 31–40, New York, NY, USA, 2008. ACM.

[115] Xindong Wu, Vipin Kumar, Quinlan J. Ross, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[116] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning with applications to clustering with side information. In *16th Conference on Advances in Neural Information Processing Systems*, 2002.

[117] Junling Xu, Baowen Xu, Weifeng Zhang, Wei Zhang, and Jun Hou. Stable initialization scheme for k-means clustering. *Wuhan University Journal of Natural Sciences*, pages 24–28, 2009.

[118] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. SCAN: a structural clustering algorithm for networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 824–833, New York, NY, USA, 2007. ACM.

[119] Liu Yang and Advisor R. Jin. Contents distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.

[120] Liu Yang, Rong Jin, Lily Mummert, Rahul Sukthankar, Adam Goode, Bin Zheng, Steven C.H. Hoi, and Mahadev Satyanarayanan. A boosting framework for visuality-preserving distance metric learning and its application to medical image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:30–44, 2010.

[121] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*, WWW '11, pages 537–546, 2011.

[122] Zijun Yang and C.-C. Jay Kuo. Survey on image content analysis, indexing, and retrieval techniques and status report of mpeg-7. *Tamkang Journal of Science and Engineering*, 3(2):101–118, 1999.

[123] Zheng Ye, Xiangji Huang, Qinmin Hu, and Hongfei Lin. An integrated approach for medical image retrieval through combining textual and visual features. In *Proceedings of the 10th international conference on Cross-language evaluation forum: multimedia experiments*, CLEF'09, pages 195–202, Berlin, Heidelberg, 2010. Springer-Verlag.

[124] Xiaoxin Yin, J. H., and Philip S. Yu. LinkClus: efficient clustering via heterogeneous semantic links. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 427–438, 2006.

[125] Kenichi Yoshida, Fuminori Adachi, Takashi Washio, Hiroshi Motoda, Teruaki Homma, Akihiro Nakashima, Hiromitsu Fujikawa, and Katsuyuki Yamazaki. Density-based spam detector. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 486–493, 2004.

[126] Jie Yu, Xin Jin, Jiawei Han, and Jiebo Luo. Collection-based sparse label propagation and its application on social group suggestion from photos. *ACM Trans. Intell. Syst. Technol.*, 2:12:1–12:21, February 2011.

[127] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *Int. J. Comput. Vision*, 73:213–238, June 2007.

[128] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114. ACM, 1996.